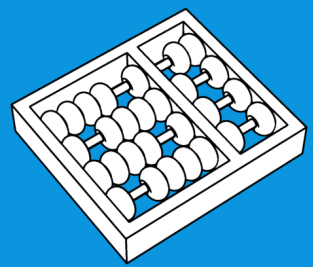
Reduções e cotas inferiores

MC558 - Projeto e Análise de Algoritmos II

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

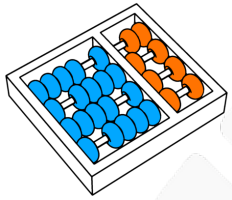
UNICAMP



"A nossa capacidade de resolver problemas é limitada pela nossa concepção do que é viável."

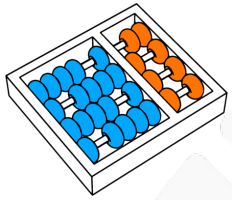
Russell L. Ackoff

Última aula



Redução

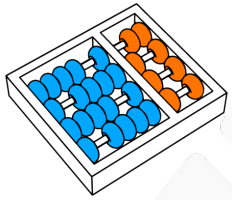
Problema A:



Redução

Problema A:

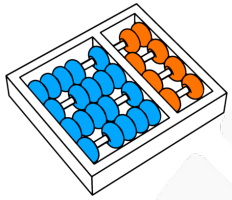
- Instância: I_A



Redução

Problema A:

- Instância: I_A
- Solução: S_A

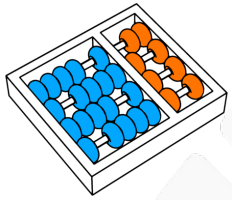


Redução

Problema A:

- Instância: I_A
- Solução: S_A

Problema B:



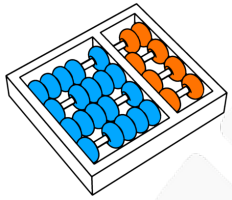
Redução

Problema A:

- Instância: I_A
- Solução: S_A

Problema B:

- Instância: I_B



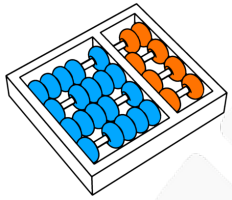
Redução

Problema A:

- Instância: I_A
- Solução: S_A

Problema B:

- Instância: I_B
- Solução: S_B



Redução

Problema A:

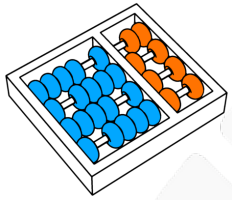
- Instância: I_A
- Solução: S_A

Problema B:

- Instância: I_B
- Solução: S_B

Definição

*Uma **REDUÇÃO** do problema A ao problema B é um par de sub-rotinas τ_I e τ_S tais que:*



Redução

Problema A:

- Instância: I_A
- Solução: S_A

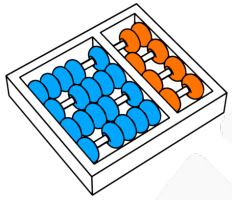
Problema B:

- Instância: I_B
- Solução: S_B

Definição

*Uma **REDUÇÃO** do problema A ao problema B é um par de sub-rotinas τ_I e τ_S tais que:*

- τ_I *transforma uma instância I_A de A em uma instância I_B de B.*



Redução

Problema A:

- Instância: I_A
- Solução: S_A

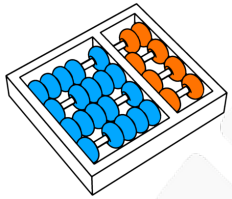
Problema B:

- Instância: I_B
- Solução: S_B

Definição

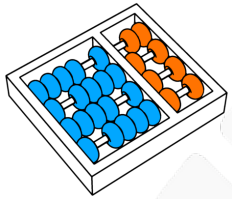
*Uma **REDUÇÃO** do problema A ao problema B é um par de sub-rotinas τ_I e τ_S tais que:*

- τ_I *transforma uma instância I_A de A em uma instância I_B de B.*
- τ_S *transforma uma solução S_B de I_B em uma solução S_A de I_A.*



Redução como um algoritmo

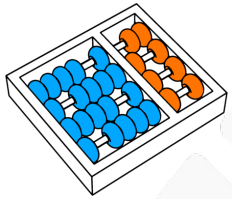
Como podemos resolver o problema A?



Redução como um algoritmo

Como podemos resolver o problema A?

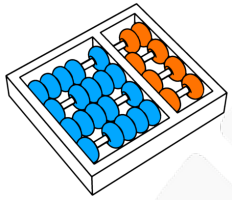
1. Suponha que existe um algoritmo $\textsc{Alg}_{\mathbf{B}}$ para o problema B.



Redução como um algoritmo

Como podemos resolver o problema A?

1. Suponha que existe um algoritmo $\textsc{Alg}_{\mathbf{B}}$ para o problema B.
2. Podemos usar $\textsc{Alg}_{\mathbf{B}}$ como uma **CAIXA-PRETA**.



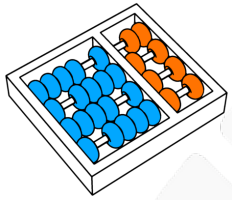
Redução como um algoritmo

Como podemos resolver o problema A?

1. Suponha que existe um algoritmo $\textsc{Alg}_{\mathbf{B}}$ para o problema B.
2. Podemos usar $\textsc{Alg}_{\mathbf{B}}$ como uma **CAIXA-PRETA**.

Algoritmo: $\textsc{Redução}(I, A)$

1 $I_B \leftarrow \tau_I(I_A)$
2 $S_B \leftarrow \textsc{Alg}_{\mathbf{B}}(\mathbf{I_B})$
3 $S_A \leftarrow \tau_S(I_A, S_B)$
4 **devolva** S_A



Redução como um algoritmo

Como podemos resolver o problema A?

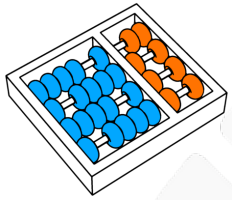
1. Suponha que existe um algoritmo $\textsc{Alg}_{\mathbf{B}}$ para o problema B.
2. Podemos usar $\textsc{Alg}_{\mathbf{B}}$ como uma **CAIXA-PRETA**.

Algoritmo: $\textsc{Redução}(I, A)$

---

1 $I_B \leftarrow \tau_I(I_A)$
2 $S_B \leftarrow \textsc{Alg}_{\mathbf{B}}(\mathbf{I_B})$
3 $S_A \leftarrow \tau_S(I_A, S_B)$
4 **devolva** S_A

Em outras palavras:



Redução como um algoritmo

Como podemos resolver o problema A?

1. Suponha que existe um algoritmo $\textsc{Alg}_{\mathbf{B}}$ para o problema B.
2. Podemos usar $\textsc{Alg}_{\mathbf{B}}$ como uma **CAIXA-PRETA**.

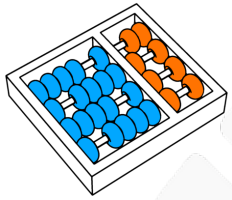
Algoritmo: $\textsc{Redução}(I, A)$

---

1 $I_B \leftarrow \tau_I(I_A)$
2 $S_B \leftarrow \textsc{Alg}_{\mathbf{B}}(\mathbf{I_B})$
3 $S_A \leftarrow \tau_S(I_A, S_B)$
4 **devolva** S_A

Em outras palavras:

- Se sabemos resolver B, então também sei resolver A!



Redução como um algoritmo

Como podemos resolver o problema A?

1. Suponha que existe um algoritmo $\textsc{Alg}_{\mathbf{B}}$ para o problema B.
2. Podemos usar $\textsc{Alg}_{\mathbf{B}}$ como uma **CAIXA-PRETA**.

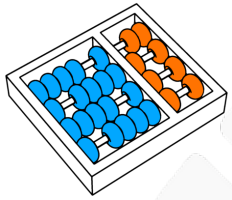
Algoritmo: $\textsc{Redução}(I, A)$

---

1 $I_B \leftarrow \tau_I(I_A)$
2 $S_B \leftarrow \textsc{Alg}_{\mathbf{B}}(\mathbf{I_B})$
3 $S_A \leftarrow \tau_S(I_A, S_B)$
4 **devolva** S_A

Em outras palavras:

- Se sabemos resolver B, então também sei resolver A!
- A é mais “fácil"que B.



Redução como um algoritmo

Como podemos resolver o problema A?

1. Suponha que existe um algoritmo $\textsc{Alg}_{\mathbf{B}}$ para o problema B.
2. Podemos usar $\textsc{Alg}_{\mathbf{B}}$ como uma **CAIXA-PRETA**.

Algoritmo: $\textsc{Redução}(I, A)$

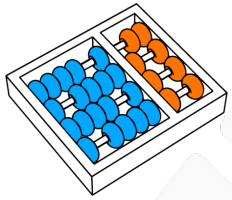
---

1 $I_B \leftarrow \tau_I(I_A)$
2 $S_B \leftarrow \textsc{Alg}_{\mathbf{B}}(\mathbf{I_B})$
3 $S_A \leftarrow \tau_S(I_A, S_B)$
4 **devolva** S_A

Em outras palavras:

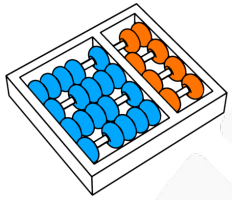
- Se sabemos resolver B, então também sei resolver A!
- A é mais “fácil"que B.
- Denotamos $\mathbf{A \preccurlyeq B}$.

Multiplicação de Matrizes para Multiplicação de Matrizes Simétricas



Considerando o caso particular

Considere um caso particular de MMQ:



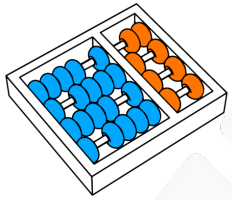
Considerando o caso particular

Considere um caso particular de MMQ:

Problema (Multiplicação de Matrizes Simétricas (MMS))

Entrada: *Uma matriz* ***SIMÉTRICA*** *quadrada A de ordem m e uma matriz* ***SIMÉTRICA*** *quadrada B de ordem m.*

Saída: *O produto $P = A \times B$.*



Considerando o caso particular

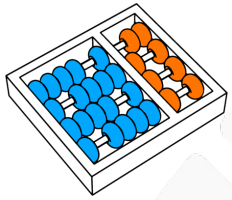
Considere um caso particular de MMQ:

Problema (Multiplicação de Matrizes Simétricas (MMS))

Entrada: *Uma matriz* ***SIMÉTRICA*** *quadrada A de ordem m e uma matriz* ***SIMÉTRICA*** *quadrada B de ordem m.*

Saída: *O produto $P = A \times B$.*

Claro que MMS $\preccurlyeq_{m^2}$ MMQ.



Considerando o caso particular

Considere um caso particular de MMQ:

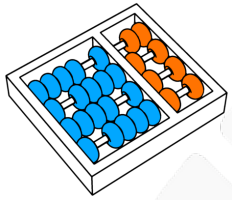
Problema (Multiplicação de Matrizes Simétricas (MMS))

Entrada: *Uma matriz* ***SIMÉTRICA*** *quadrada A de ordem m e uma matriz* ***SIMÉTRICA*** *quadrada B de ordem m.*

Saída: *O produto $P = A \times B$.*

Claro que MMS $\preccurlyeq_{m^2}$ MMQ.

- Portanto, MMQ é pelo menos tão difícil quanto MMS.



Considerando o caso particular

Considere um caso particular de MMQ:

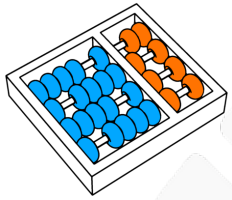
Problema (Multiplicação de Matrizes Simétricas (MMS))

Entrada: *Uma matriz* ***SIMÉTRICA*** *quadrada A de ordem m e uma matriz* ***SIMÉTRICA*** *quadrada B de ordem m.*

Saída: *O produto $P = A \times B$.*

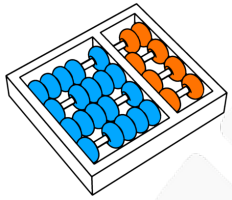
Claro que MMS $\preccurlyeq_{m^2}$ MMQ.

- Portanto, MMQ é pelo menos tão difícil quanto MMS.
- Será que MMS também é pelo menos tão difícil quanto MMQ?



Reduzindo MMQ $\propto$ MMS

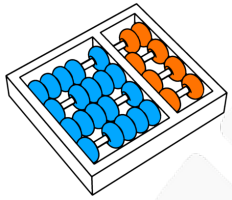
1. Considere uma instância de MMQ, $I_{MMQ} = (A, B, n)$.



Reduzindo MMQ $\propto$ MMS

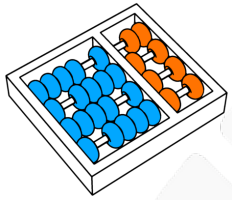
1. Considere uma instância de MMQ, $I_{MMQ} = (A, B, n)$.
2. Construa uma instância de MMS, $I_{MMS} = (A', B', 2n)$, em que

$$A' = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \quad \text{e} \quad B' = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}.$$



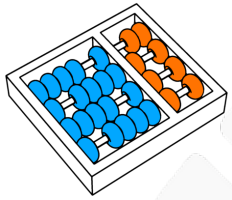
Reduzindo MMQ $\propto$ MMS

1. Considere uma instância de MMQ, $I_{MMQ} = (A, B, n)$.
2. Construa uma instância de MMS, $I_{MMS} = (A', B', 2n)$, em que
$$A' = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \quad \text{e} \quad B' = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}.$$
3. A solução de MMS é:
$$P' = A'B' = \begin{bmatrix} AB & 0 \\ 0 & A^T B^T \end{bmatrix}.$$



Reduzindo MMQ $\propto$ MMS

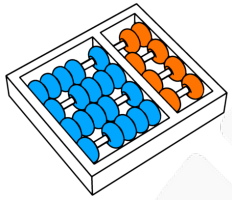
1. Considere uma instância de MMQ, $I_{MMQ} = (A, B, n)$.
2. Construa uma instância de MMS, $I_{MMS} = (A', B', 2n)$, em que
$$A' = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \quad \text{e} \quad B' = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}.$$
3. A solução de MMS é:
$$P' = A'B' = \begin{bmatrix} AB & 0 \\ 0 & A^T B^T \end{bmatrix}.$$
4. Devolva o primeiro bloco da matriz P'.



Reduzindo MMQ $\propto$ MMS

1. Considere uma instância de MMQ, $I_{MMQ} = (A, B, n)$.
2. Construa uma instância de MMS, $I_{MMS} = (A', B', 2n)$, em que
$$A' = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \quad \text{e} \quad B' = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}.$$
3. A solução de MMS é:
$$P' = A'B' = \begin{bmatrix} AB & 0 \\ 0 & A^T B^T \end{bmatrix}.$$
4. Devolva o primeiro bloco da matriz P'.

Tempo da redução: $O(n^2)$.

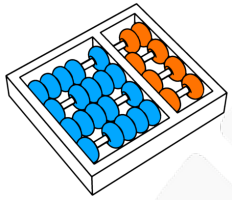


Reduzindo MMQ $\propto$ MMS

1. Considere uma instância de MMQ, $I_{MMQ} = (A, B, n)$.
2. Construa uma instância de MMS, $I_{MMS} = (A', B', 2n)$, em que
$$A' = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \quad \text{e} \quad B' = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}.$$
3. A solução de MMS é:
$$P' = A'B' = \begin{bmatrix} AB & 0 \\ 0 & A^T B^T \end{bmatrix}.$$
4. Devolva o primeiro bloco da matriz P'.

Tempo da redução: $O(n^2)$.

- Construir I_{MMQ} leva tempo $O(n^2)$.

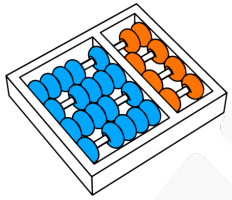


Reduzindo MMQ $\propto$ MMS

1. Considere uma instância de MMQ, $I_{MMQ} = (A, B, n)$.
2. Construa uma instância de MMS, $I_{MMS} = (A', B', 2n)$, em que
$$A' = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \quad \text{e} \quad B' = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}.$$
3. A solução de MMS é:
$$P' = A'B' = \begin{bmatrix} AB & 0 \\ 0 & A^T B^T \end{bmatrix}.$$
4. Devolva o primeiro bloco da matriz P'.

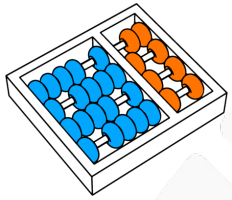
Tempo da redução: $O(n^2)$.

- Construir I_{MMQ} leva tempo $O(n^2)$.
- Copiar o bloco e P' leva tempo $O(n^2)$.



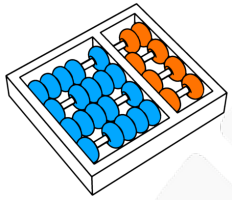
Interpretando os fatos

- Suponha que exista um algoritmo para MMS de tempo $\mathbf{O(T(m))}$ para algum polinômio $T(m)$.



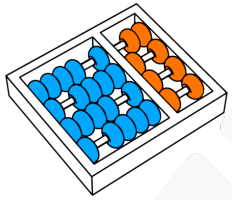
Interpretando os fatos

- Suponha que exista um algoritmo para MMS de tempo $\mathbf{O(T(m))}$ para algum polinômio $T(m)$.
  - Lembre que m é a ordem das matrizes A' e B'.



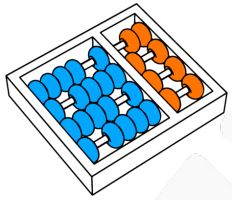
Interpretando os fatos

- Suponha que exista um algoritmo para MMS de tempo **O(T(m))** para algum polinômio $T(m)$.
  - Lembre que m é a ordem das matrizes A' e B'.
- Quão pequeno pode ser $T(m)$?



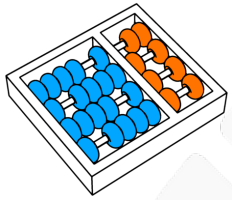
Interpretando os fatos

- Suponha que exista um algoritmo para MMS de tempo **O(T(m))** para algum polinômio $T(m)$.
  - Lembre que m é a ordem das matrizes A' e B'.
- Quão pequeno pode ser $T(m)$?
  - É claro que $T(m) = \Omega(m^2)$, pois é preciso ler a entrada.



Interpretando os fatos

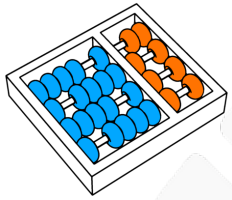
- Suponha que exista um algoritmo para MMS de tempo **O(T(m))** para algum polinômio $T(m)$.
  - Lembre que m é a ordem das matrizes A' e B'.
- Quão pequeno pode ser $T(m)$?
  - É claro que $T(m) = \Omega(m^2)$, pois é preciso ler a entrada.
  - Será que pode ser mais rápido que $o(m^{2,3728639})$?



Interpretando os fatos

- Suponha que exista um algoritmo para MMS de tempo **O(T(m))** para algum polinômio $T(m)$.
  - Lembre que m é a ordem das matrizes A' e B'.
- Quão pequeno pode ser $T(m)$?
  - É claro que $T(m) = \Omega(m^2)$, pois é preciso ler a entrada.
  - Será que pode ser mais rápido que $o(m^{2,3728639})$?

Para responder isso, usamos a redução MMQ $\preccurlyeq_{n^2}$ MMS:

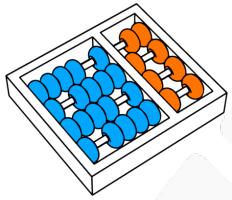


Interpretando os fatos

- Suponha que exista um algoritmo para MMS de tempo **O(T(m))** para algum polinômio $T(m)$.
  - Lembre que m é a ordem das matrizes A' e B'.
- Quão pequeno pode ser $T(m)$?
  - É claro que $T(m) = \Omega(m^2)$, pois é preciso ler a entrada.
  - Será que pode ser mais rápido que $o(m^{2,3728639})$?

Para responder isso, usamos a redução MMQ $\preccurlyeq_{n^2}$ MMS:

- Ela implica em um algoritmo de tempo total $O(T(m) + n^2)$.

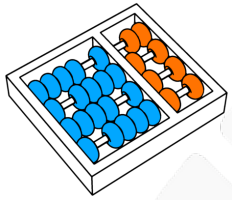


Interpretando os fatos

- Suponha que exista um algoritmo para MMS de tempo **O(T(m))** para algum polinômio $T(m)$.
  - Lembre que m é a ordem das matrizes A' e B'.
- Quão pequeno pode ser $T(m)$?
  - É claro que $T(m) = \Omega(m^2)$, pois é preciso ler a entrada.
  - Será que pode ser mais rápido que $o(m^{2,3728639})$?

Para responder isso, usamos a redução MMQ $\preccurlyeq_{n^2}$ MMS:

- Ela implica em um algoritmo de tempo total $O(T(m) + n^2)$.
- Como $m = 2n$, tempo é $O(T(2n) + n^2) = O(T(n) + n^2)$.

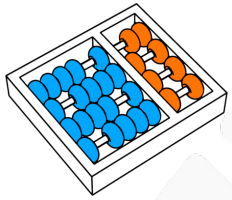


Interpretando os fatos

- Suponha que exista um algoritmo para MMS de tempo **O(T(m))** para algum polinômio $T(m)$.
  - Lembre que m é a ordem das matrizes A' e B'.
- Quão pequeno pode ser $T(m)$?
  - É claro que $T(m) = \Omega(m^2)$, pois é preciso ler a entrada.
  - Será que pode ser mais rápido que $o(m^{2,3728639})$?

Para responder isso, usamos a redução MMQ $\preccurlyeq_{n^2}$ MMS:

- Ela implica em um algoritmo de tempo total $O(T(m) + n^2)$.
- Como $m = 2n$, tempo é $O(T(2n) + n^2) = O(T(n) + n^2)$.
- Como $T(n)$ domina n^2, o tempo é simplesmente $O(T(n))$.



Interpretando os fatos

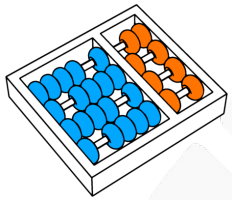
- Suponha que exista um algoritmo para MMS de tempo **O(T(m))** para algum polinômio $T(m)$.
  - Lembre que m é a ordem das matrizes A' e B'.
- Quão pequeno pode ser $T(m)$?
  - É claro que $T(m) = \Omega(m^2)$, pois é preciso ler a entrada.
  - Será que pode ser mais rápido que $o(m^{2,3728639})$?

Para responder isso, usamos a redução MMQ $\preccurlyeq_{n^2}$ MMS:

- Ela implica em um algoritmo de tempo total $O(T(m) + n^2)$.
- Como $m = 2n$, tempo é $O(T(2n) + n^2) = O(T(n) + n^2)$.
- Como $T(n)$ domina n^2, o tempo é simplesmente $O(T(n))$.

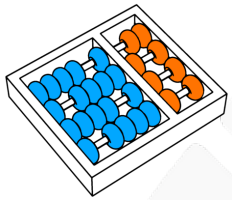
Ou seja: Um algoritmo com complexidade $T(m)$ para MMS implica em um algoritmo com complexidade $T(n)$ para MMQ!

Reduções para obtenção de cota inferior



Uma redução $A \preccurlyeq_{f(n)} B$

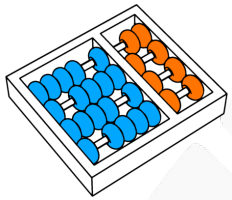
Suponha que:



Uma redução $A \preccurlyeq_{f(n)} B$

Suponha que:

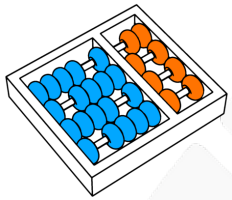
- A tem cota inferior $h(n)$.



Uma redução $A \preccurlyeq_{f(n)} B$

Suponha que:

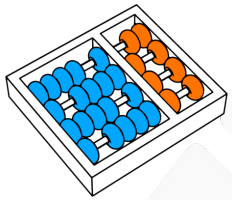
- A tem cota inferior $h(n)$.
- $\textsc{Alg}_B$ resolve B em tempo $g(n)$.



Uma redução $A \preccurlyeq_{f(n)} B$

Suponha que:

- A tem cota inferior $h(n)$.
- $\textsc{Alg}_B$ resolve B em tempo $g(n)$.
- A redução gasta tempo $f(n) \leq \frac{h(n)}{2}$.

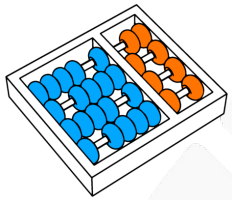


Uma redução $A \preccurlyeq_{f(n)} B$

Suponha que:

- A tem cota inferior $h(n)$.
- $\textsc{Alg}_B$ resolve B em tempo $g(n)$.
- A redução gasta tempo $f(n) \leq \frac{h(n)}{2}$.

Então, um algoritmo baseado na redução $A \preccurlyeq_{f(n)} B$ tem tmepo:



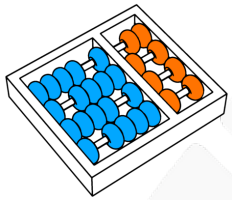
Uma redução $A \preccurlyeq_{f(n)} B$

Suponha que:

- A tem cota inferior $h(n)$.
- $\textsc{Alg}_B$ resolve B em tempo $g(n)$.
- A redução gasta tempo $f(n) \leq \frac{h(n)}{2}$.

Então, um algoritmo baseado na redução $A \preccurlyeq_{f(n)} B$ tem tmepo:

$f(n) + g(n)$



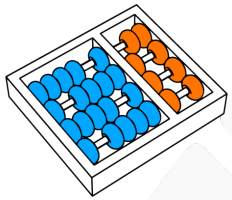
Uma redução $A \preccurlyeq_{f(n)} B$

Suponha que:

- A tem cota inferior $h(n)$.
- $\textsc{Alg}_B$ resolve B em tempo $g(n)$.
- A redução gasta tempo $f(n) \leq \frac{h(n)}{2}$.

Então, um algoritmo baseado na redução $A \preccurlyeq_{f(n)} B$ tem tmepo:

$$f(n) + g(n) \geq h(n)$$



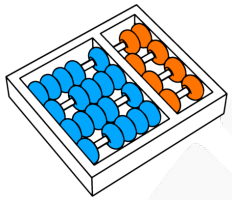
Uma redução $A \preccurlyeq_{f(n)} B$

Suponha que:

- A tem cota inferior $h(n)$.
- $\textsc{Alg}_B$ resolve B em tempo $g(n)$.
- A redução gasta tempo $f(n) \leq \frac{h(n)}{2}$.

Então, um algoritmo baseado na redução $A \preccurlyeq_{f(n)} B$ tem tmepo:

$f(n) + g(n) \geq h(n) \Rightarrow$



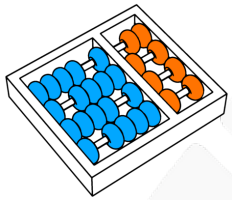
Uma redução $A \preccurlyeq_{f(n)} B$

Suponha que:

- A tem cota inferior $h(n)$.
- ALG_B resolve B em tempo $g(n)$.
- A redução gasta tempo $f(n) \leq \frac{h(n)}{2}$.

Então, um algoritmo baseado na redução $A \preccurlyeq_{f(n)} B$ tem tmepo:

$$f(n) + g(n) \geq h(n) \Rightarrow g(n) \geq h(n) - f(n)$$



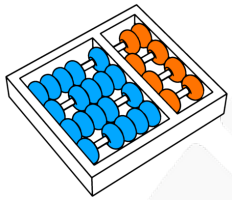
Uma redução $A \preccurlyeq_{f(n)} B$

Suponha que:

- A tem cota inferior $h(n)$.
- ALG_B resolve B em tempo $g(n)$.
- A redução gasta tempo $f(n) \leq \frac{h(n)}{2}$.

Então, um algoritmo baseado na redução $A \preccurlyeq_{f(n)} B$ tem tmepo:

$$f(n) + g(n) \geq h(n) \Rightarrow g(n) \geq h(n) - f(n) \geq h(n) - \frac{h(n)}{2}$$



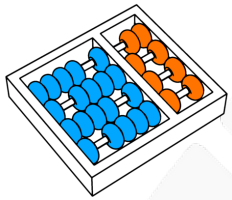
Uma redução $A \preccurlyeq_{f(n)} B$

Suponha que:

- A tem cota inferior $h(n)$.
- ALG_B resolve B em tempo $g(n)$.
- A redução gasta tempo $f(n) \leq \frac{h(n)}{2}$.

Então, um algoritmo baseado na redução $A \preccurlyeq_{f(n)} B$ tem tmepo:

$$f(n) + g(n) \geq h(n) \Rightarrow g(n) \geq h(n) - f(n) \geq h(n) - \frac{h(n)}{2} =$$



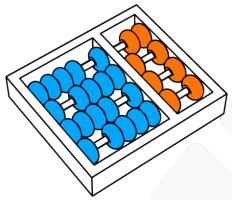
Uma redução $A \preccurlyeq_{f(n)} B$

Suponha que:

- A tem cota inferior $h(n)$.
- ALG_B resolve B em tempo $g(n)$.
- A redução gasta tempo $f(n) \leq \frac{h(n)}{2}$.

Então, um algoritmo baseado na redução $A \preccurlyeq_{f(n)} B$ tem tmepo:

$$f(n) + g(n) \geq h(n) \Rightarrow g(n) \geq h(n) - f(n) \geq h(n) - \frac{h(n)}{2} = \frac{h(n)}{2}$$



Uma redução $A \preccurlyeq_{f(n)} B$

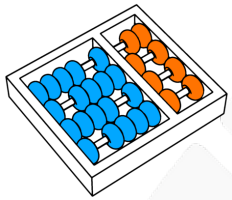
Suponha que:

- A tem cota inferior $h(n)$.
- ALG_B resolve B em tempo $g(n)$.
- A redução gasta tempo $f(n) \leq \frac{h(n)}{2}$.

Então, um algoritmo baseado na redução $A \preccurlyeq_{f(n)} B$ tem tmepo:

$$f(n) + g(n) \geq h(n) \Rightarrow g(n) \geq h(n) - f(n) \geq h(n) - \frac{h(n)}{2} = \frac{h(n)}{2}$$

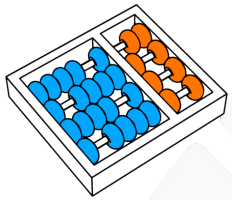
Conclusão: $g(n) \geq \Omega(h(n))$.



Transferindo cotas inferiores

Teorema

Considere dois problemas A e B e suponha que

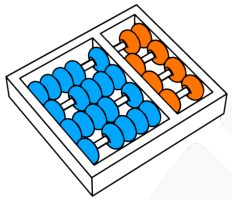


Transferindo cotas inferiores

Teorema

Considere dois problemas A e B e suponha que

1. *$h(n)$ é cota inferior para A e*

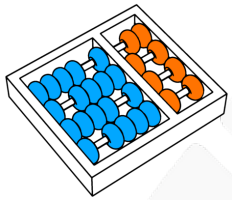


Transferindo cotas inferiores

Teorema

Considere dois problemas A e B e suponha que

1. *$h(n)$ é cota inferior para A e*
2. *$A \preccurlyeq_{f(n)} B$,*

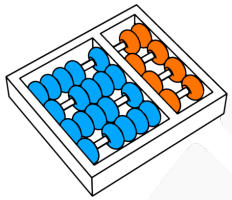


Transferindo cotas inferiores

Teorema

Considere dois problemas A e B e suponha que

1. $h(n)$ *é cota inferior para* A *e*
2. $A \preccurlyeq_{f(n)} B$,
3. $f(n) = o(h(n))$.



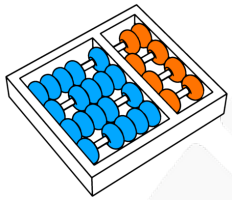
Transferindo cotas inferiores

Teorema

Considere dois problemas A e B e suponha que

1. *$h(n)$ é cota inferior para A e*
2. *$A \preccurlyeq_{f(n)} B$,*
3. *$f(n) = o(h(n))$.*

Então $h(n)$ é cota inferior para B.



Transferindo cotas inferiores

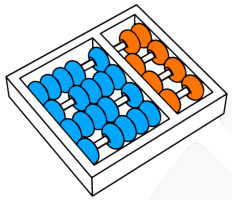
Teorema

Considere dois problemas A e B e suponha que

1. *$h(n)$ é cota inferior para A e*
2. *$A \preccurlyeq_{f(n)} B$,*
3. *$f(n) = o(h(n))$.*

Então $h(n)$ é cota inferior para B.

Observações:



Transferindo cotas inferiores

Teorema

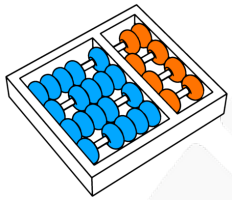
Considere dois problemas A e B e suponha que

1. *$h(n)$ é cota inferior para A e*
2. *$A \preccurlyeq_{f(n)} B$,*
3. *$f(n) = o(h(n))$.*

Então $h(n)$ é cota inferior para B.

Observações:

- A cota inferior depende do **MODELO DE COMPUTAÇÃO**.



Transferindo cotas inferiores

Teorema

Considere dois problemas A e B e suponha que

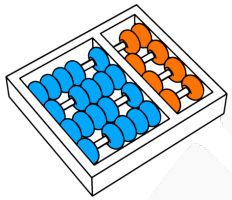
1. *$h(n)$ é cota inferior para A e*
2. *$A \preccurlyeq_{f(n)} B$,*
3. *$f(n) = o(h(n))$.*

Então $h(n)$ é cota inferior para B.

Observações:

- A cota inferior depende do **MODELO DE COMPUTAÇÃO**.
- Supomos o mesmo modelo para ambos problemas.

Ordenação para Envoltória Convexa

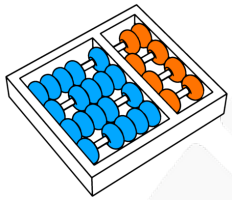


Problema de origem

Problema (Ordenação (ORD))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *elementos comparáveis.*

Saída: *Uma permutação de* X *cujos elementos estejam ordenados.*



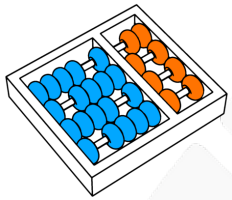
Problema de origem

Problema (Ordenação (ORD))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *elementos comparáveis.*

Saída: *Uma permutação de* X *cujos elementos estejam ordenados.*

Observações:



Problema de origem

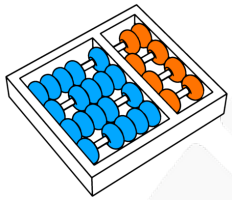
Problema (Ordenação (ORD))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *elementos comparáveis.*

Saída: *Uma permutação de* X *cujos elementos estejam ordenados.*

Observações:

- Só podemos comparar dois elementos por uma sub-rotina caixa-preta de tempo constante (chamada **ORÁCULO**).



Problema de origem

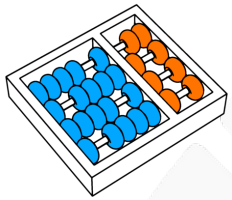
Problema (Ordenação (ORD))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *elementos comparáveis.*

Saída: *Uma permutação de* X *cujos elementos estejam ordenados.*

Observações:

- Só podemos comparar dois elementos por uma sub-rotina caixa-preta de tempo constante (chamada **ORÁCULO**).
- Esse problema tem cota inferior $\Omega(n \log n)$.

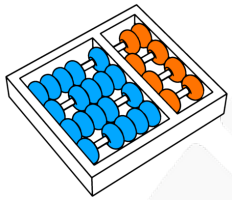


Problema de destino

Problema (Envoltória Convexa (EC))

Entrada: *Um conjunto* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de n pontos no plano.*

Saída: ***MENOR POLÍGONO CONVEXO*** *que contém os n pontos.*

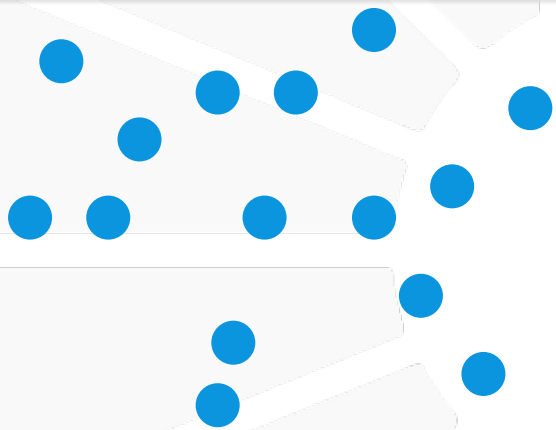


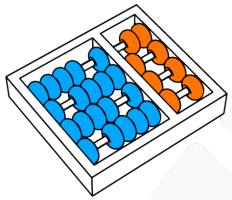
Problema de destino

Problema (Envoltória Convexa (EC))

Entrada: *Um conjunto* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de n pontos no plano.*

Saída: ***MENOR POLÍGONO CONVEXO*** *que contém os n pontos.*



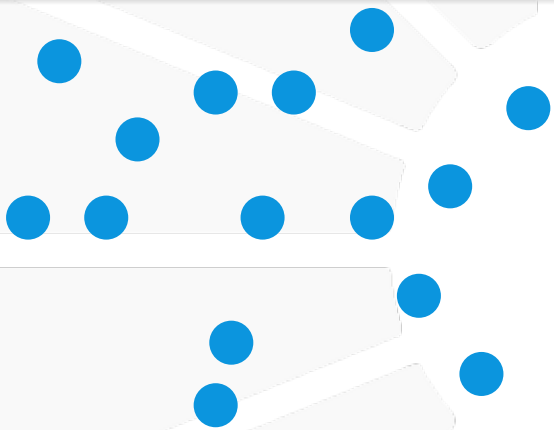


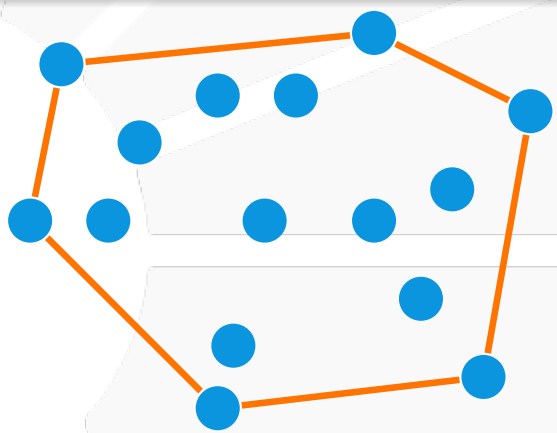
Problema de destino

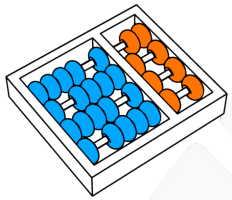
Problema (Envoltória Convexa (EC))

Entrada: *Um conjunto* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de n pontos no plano.*

Saída: ***MENOR POLÍGONO CONVEXO*** *que contém os n pontos.*





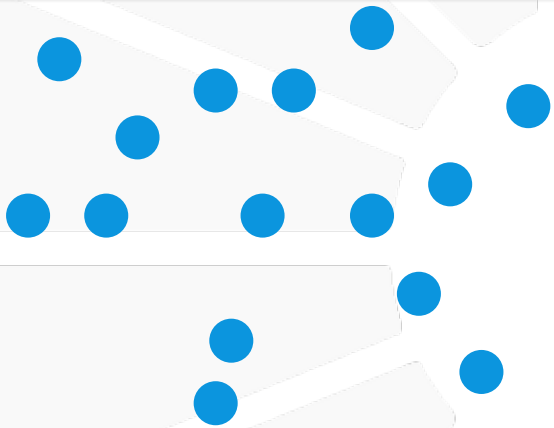


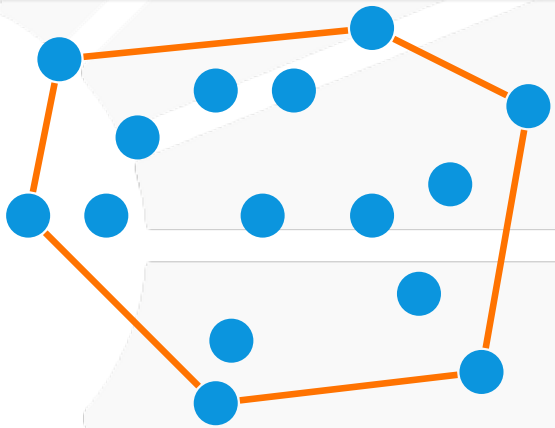
Problema de destino

Problema (Envoltória Convexa (EC))

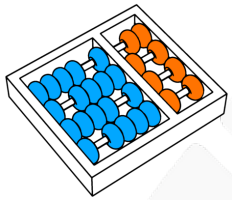
Entrada: *Um conjunto* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de n pontos no plano.*

Saída: ***MENOR POLÍGONO CONVEXO*** *que contém os n pontos.*





Observações:

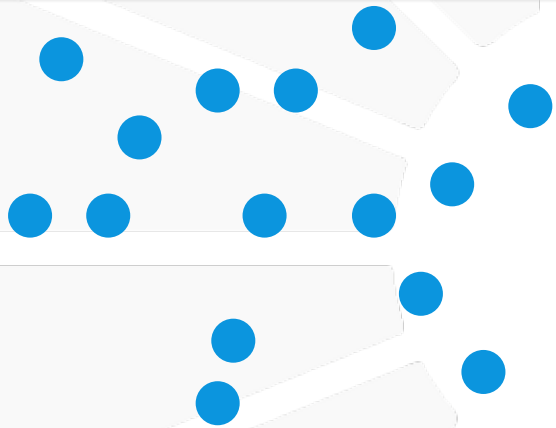


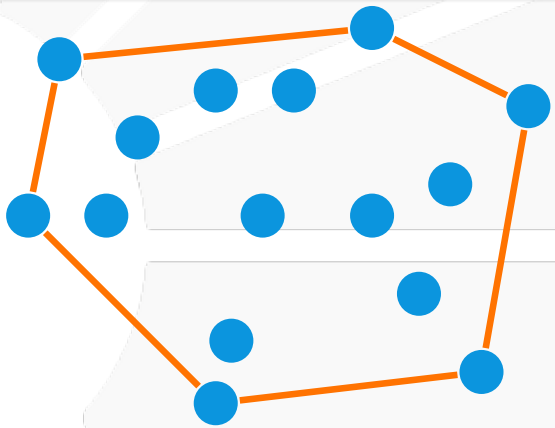
Problema de destino

Problema (Envoltória Convexa (EC))

Entrada: *Um conjunto* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de n pontos no plano.*

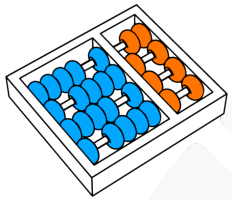
Saída: ***MENOR POLÍGONO CONVEXO*** *que contém os n pontos.*





Observações:

- Os vértices são representados em ordem anti-horária.

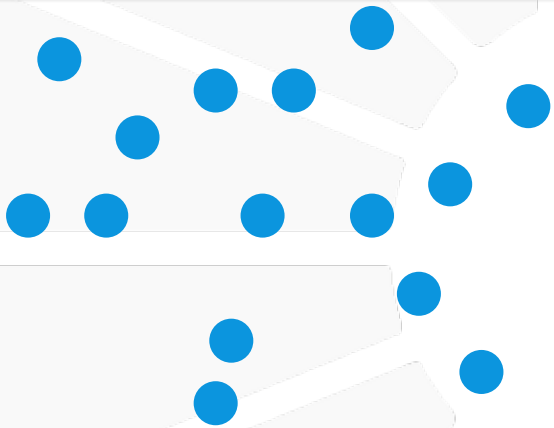


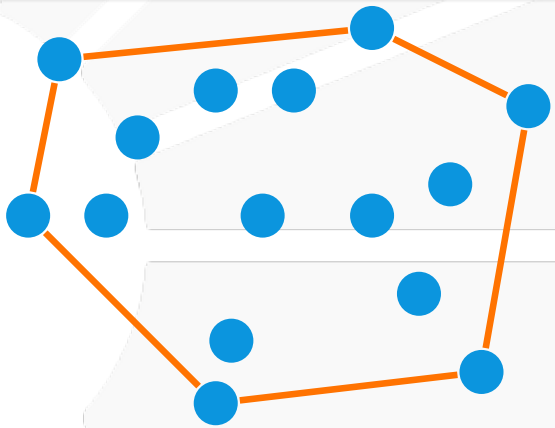
Problema de destino

Problema (Envoltória Convexa (EC))

Entrada: *Um conjunto* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de n pontos no plano.*

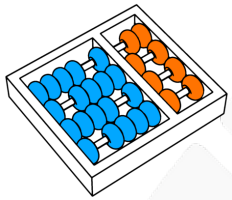
Saída: ***MENOR POLÍGONO CONVEXO*** *que contém os n pontos.*





Observações:

- Os vértices são representados em ordem anti-horária.
- Problema clássico de Geometria Computacional.

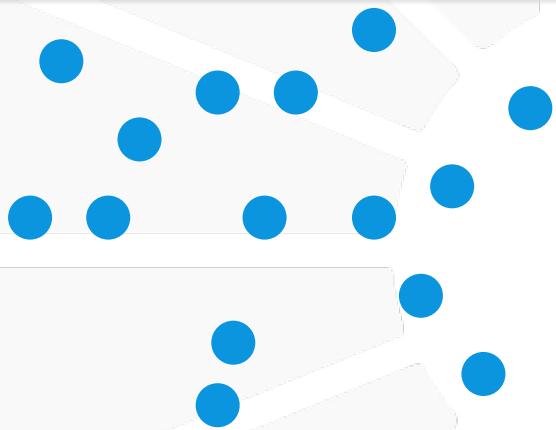


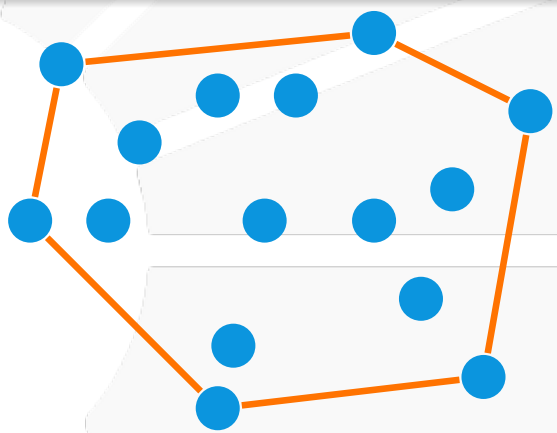
Problema de destino

Problema (Envoltória Convexa (EC))

Entrada: *Um conjunto* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de* n *pontos no plano.*

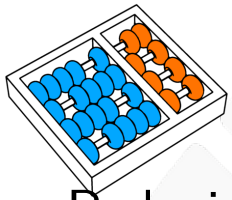
Saída: ***MENOR POLÍGONO CONVEXO*** *que contém os* n *pontos.*





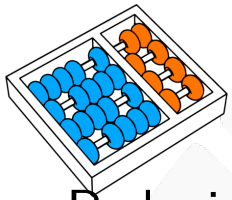
Observações:

- Os vértices são representados em ordem anti-horária.
- Problema clássico de Geometria Computacional.
- Pode ser resolvido em tempo $O(n \log n)$.



ORD $\preceq_n$ EC

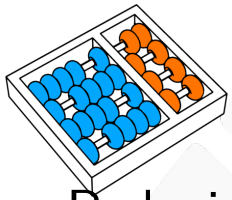
Reduzindo ORD $\preceq_n$ EC:



ORD $\preccurlyeq_n$ EC

Reduzindo ORD $\preccurlyeq_n$ EC:

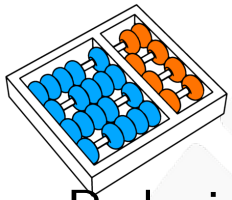
1. Considere uma instância de $I_{ORD} = (x_1, x_2 \ldots, x_n)$.



ORD $\preccurlyeq_n$ EC

Reduzindo ORD $\preccurlyeq_n$ EC:

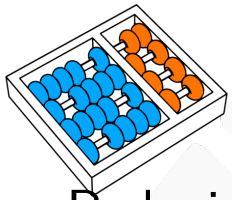
1. Considere uma instância de $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
2. Construa instância $I_{EC} = \{(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)\}$.



ORD $\preccurlyeq_n$ EC

Reduzindo ORD $\preccurlyeq_n$ EC:

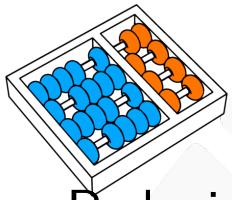
1. Considere uma instância de $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
2. Construa instância $I_{EC} = \{(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)\}$.



ORD $\preccurlyeq_n$ EC

Reduzindo ORD $\preccurlyeq_n$ EC:

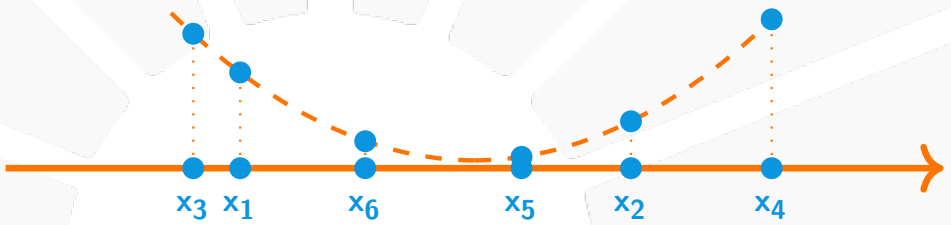
1. Considere uma instância de $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
2. Construa instância $I_{EC} = \{(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)\}$.

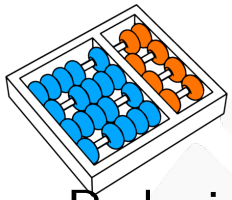


ORD $\preccurlyeq_n$ EC

Reduzindo ORD $\preccurlyeq_n$ EC:

1. Considere uma instância de $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
2. Construa instância $I_{EC} = \{(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)\}$.

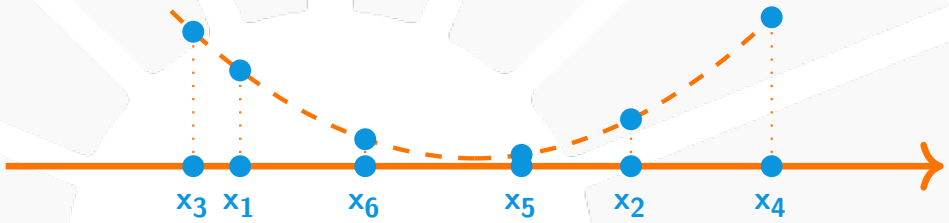




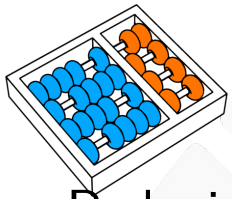
ORD $\preccurlyeq_n$ EC

Reduzindo ORD $\preccurlyeq_n$ EC:

1. Considere uma instância de $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
2. Construa instância $I_{EC} = \{(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)\}$.



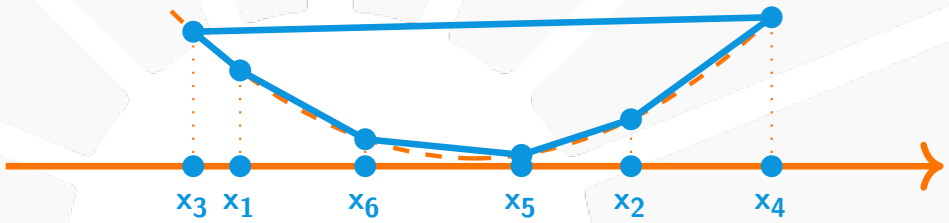
3. Resolva I_{EC} e obtenha solução S_{EC}, que é uma lista **CÍCLICA** dos vértices do polígono.



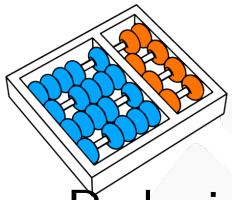
ORD $\preccurlyeq_n$ EC

Reduzindo ORD $\preccurlyeq_n$ EC:

1. Considere uma instância de $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
2. Construa instância $I_{EC} = \{(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)\}$.



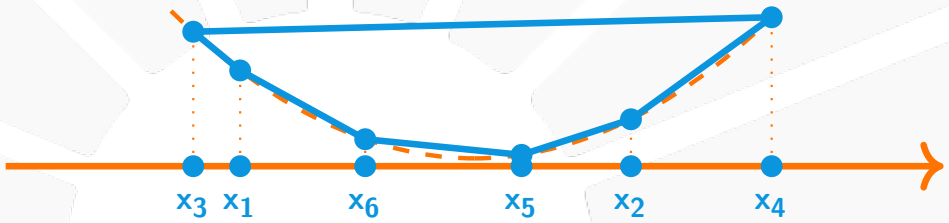
3. Resolva I_{EC} e obtenha solução S_{EC}, que é uma lista **CÍCLICA** dos vértices do polígono.



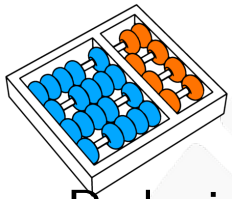
ORD $\preccurlyeq_n$ EC

Reduzindo ORD $\preccurlyeq_n$ EC:

1. Considere uma instância de $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
2. Construa instância $I_{EC} = \{(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)\}$.



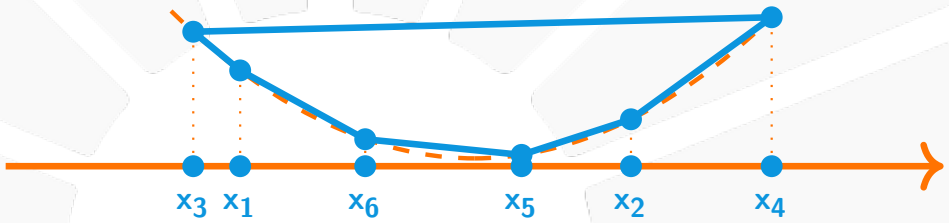
3. Resolva I_{EC} e obtenha solução S_{EC}, que é uma lista **CÍCLICA** dos vértices do polígono.
4. Determine índice i de S_{EC} do ponto com menor abcissa.



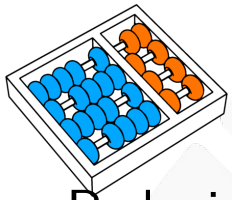
ORD $\preccurlyeq_n$ EC

Reduzindo ORD $\preccurlyeq_n$ EC:

1. Considere uma instância de $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
2. Construa instância $I_{EC} = \{(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)\}$.



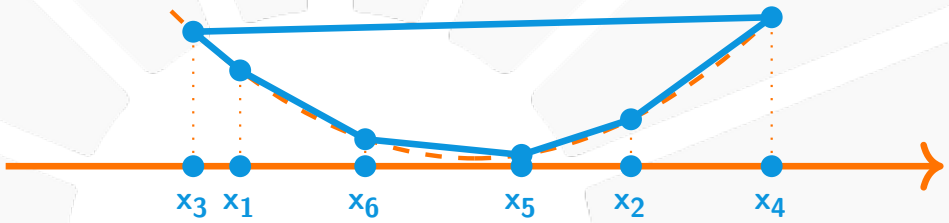
3. Resolva I_{EC} e obtenha solução S_{EC}, que é uma lista **CÍCLICA** dos vértices do polígono.
4. Determine índice i de S_{EC} do ponto com menor abcissa.
5. Liste os todos os índices a partir de i.



ORD $\preccurlyeq_n$ EC

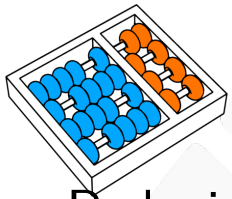
Reduzindo ORD $\preccurlyeq_n$ EC:

1. Considere uma instância de $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
2. Construa instância $I_{EC} = \{(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)\}$.



3. Resolva I_{EC} e obtenha solução S_{EC}, que é uma lista **CÍCLICA** dos vértices do polígono.
4. Determine índice i de S_{EC} do ponto com menor abcissa.
5. Liste os todos os índices a partir de i.

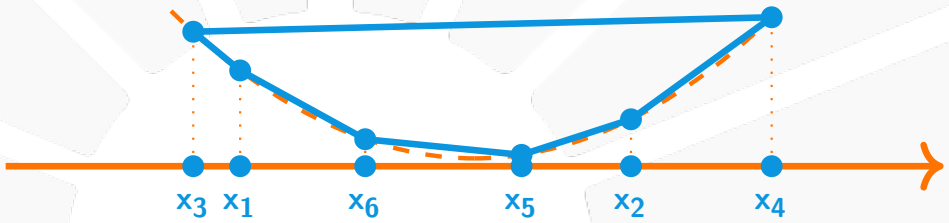
- O tempo da redução é $O(n)$.



ORD $\preccurlyeq_n$ EC

Reduzindo ORD $\preccurlyeq_n$ EC:

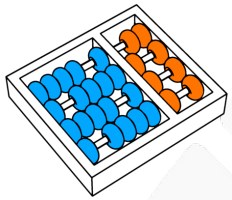
1. Considere uma instância de $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
2. Construa instância $I_{EC} = \{(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)\}$.



3. Resolva I_{EC} e obtenha solução S_{EC}, que é uma lista **CÍCLICA** dos vértices do polígono.
4. Determine índice i de S_{EC} do ponto com menor abcissa.
5. Liste os todos os índices a partir de i.

- O tempo da redução é $O(n)$.
- Portanto $\Omega(n \log n)$ também é **COTA INFERIOR** para EC.

Unicidade de Elementos para Ponto Mais Próximo em 2D

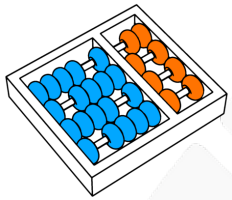


Problema de origem

Problema (Unicidade de Elementos (UE))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *elementos comparáveis.*

Saída: *Decidir se os elementos são* ***TODOS*** *distintos.*



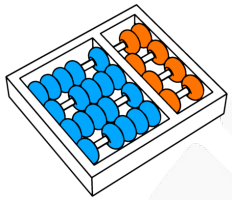
Problema de origem

Problema (Unicidade de Elementos (UE))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *elementos comparáveis.*

Saída: *Decidir se os elementos são* ***TODOS*** *distintos.*

Observações:



Problema de origem

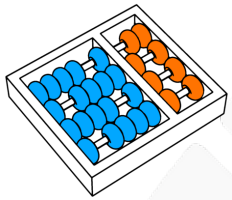
Problema (Unicidade de Elementos (UE))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *elementos comparáveis.*

Saída: *Decidir se os elementos são* ***TODOS*** *distintos.*

Observações:

- Só podemos comparar dois elementos por uma sub-rotina caixa-preta de tempo constante (chamada **ORÁCULO**).



Problema de origem

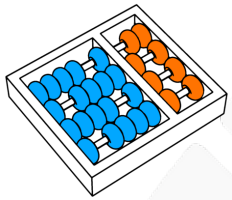
Problema (Unicidade de Elementos (UE))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *elementos comparáveis.*

Saída: *Decidir se os elementos são* ***TODOS*** *distintos.*

Observações:

- Só podemos comparar dois elementos por uma sub-rotina caixa-preta de tempo constante (chamada **ORÁCULO**).
- Esse problema tem cota inferior $\Omega(n \log n)$.



Problema de origem

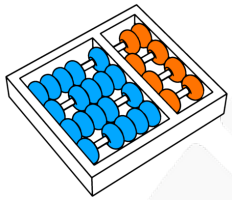
Problema (Unicidade de Elementos (UE))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *elementos comparáveis.*

Saída: *Decidir se os elementos são* ***TODOS*** *distintos.*

Observações:

- Só podemos comparar dois elementos por uma sub-rotina caixa-preta de tempo constante (chamada **ORÁCULO**).
- Esse problema tem cota inferior $\Omega(n \log n)$.
- O problema pode ser resolvido em tempo $O(n \log n)$ (como?).

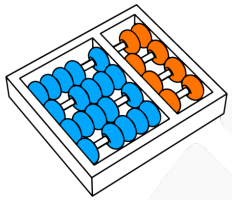


Problema de destino

Problema (Par Mais Próximo (PMP))

Entrada: *Uma coleção* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de* n *pontos no plano.*

Saída: *Par de pontos* i *e* j *que estejam* ***A MENOR DISTÂNCIA****.*

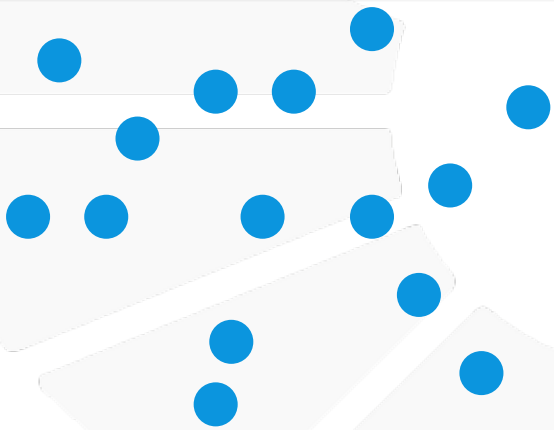


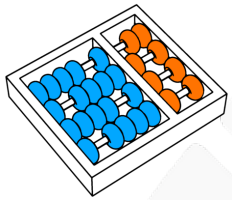
Problema de destino

Problema (Par Mais Próximo (PMP))

Entrada: *Uma coleção* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de* n *pontos no plano.*

Saída: *Par de pontos* i *e* j *que estejam* ***A MENOR DISTÂNCIA****.*



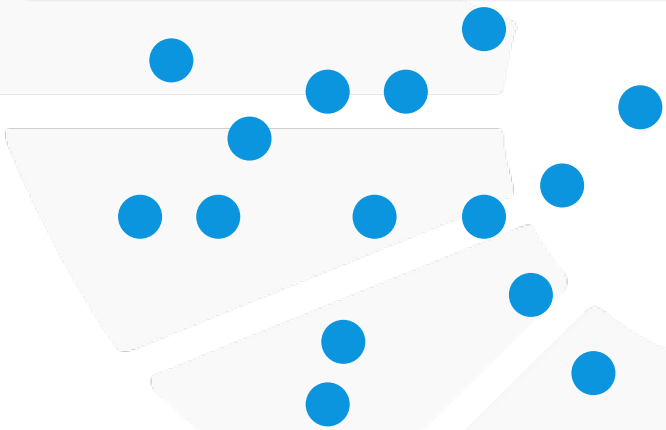


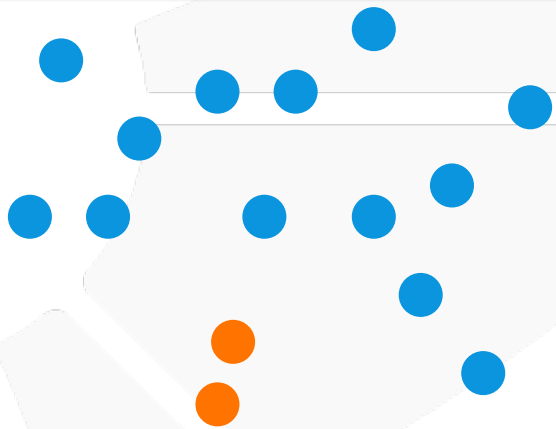
Problema de destino

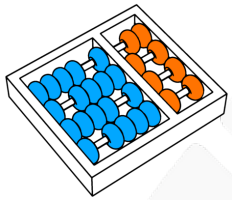
Problema (Par Mais Próximo (PMP))

Entrada: *Uma coleção* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de* n *pontos no plano.*

Saída: *Par de pontos* i *e* j *que estejam* ***A MENOR DISTÂNCIA****.*





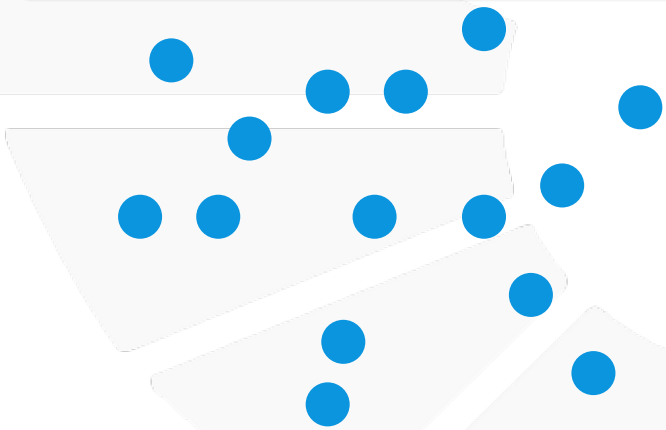


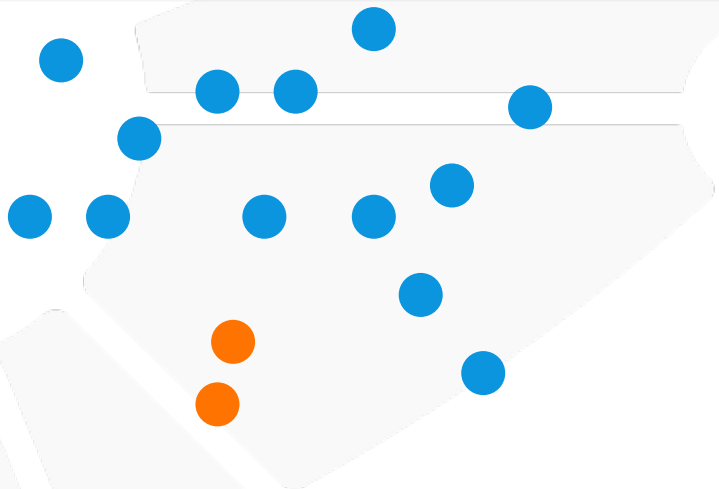
Problema de destino

Problema (Par Mais Próximo (PMP))

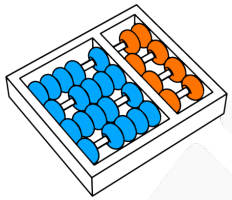
Entrada: *Uma coleção* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de* n *pontos no plano.*

Saída: *Par de pontos* i *e* j *que estejam* ***A MENOR DISTÂNCIA****.*





Observação:

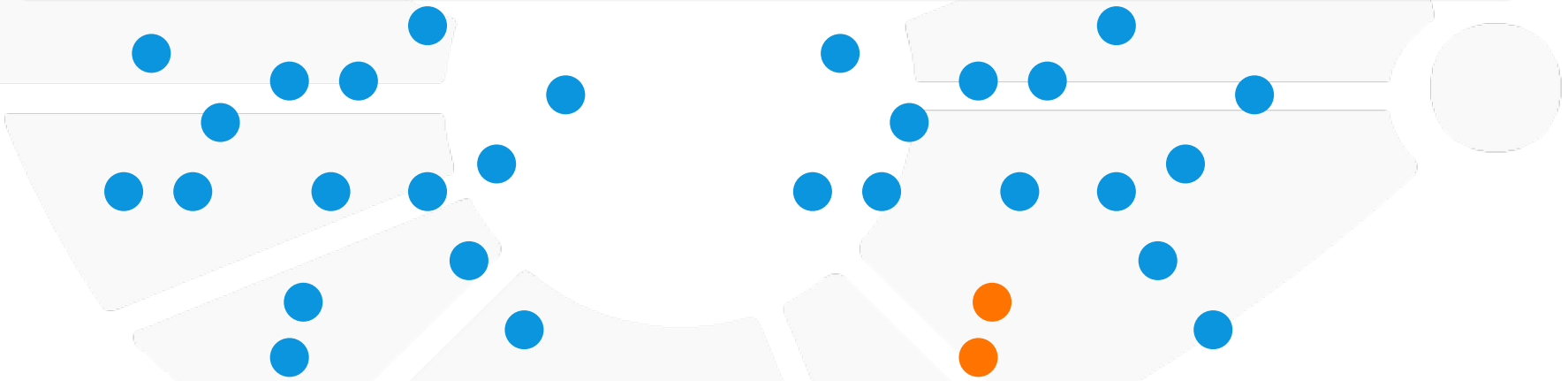


Problema de destino

Problema (Par Mais Próximo (PMP))

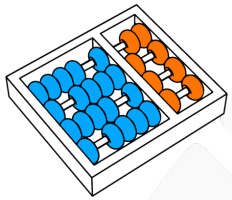
Entrada: *Uma coleção* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de* n *pontos no plano.*

Saída: *Par de pontos* i *e* j *que estejam* ***A MENOR DISTÂNCIA****.*



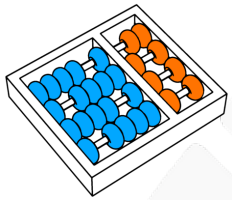
Observação:

- Pode ser resolvido em tempo $O(n \log n)$



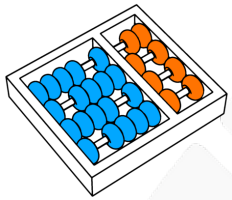
Reduzindo UE $\propto_n$ PMP

1. Considere instância $I_{UE} = (x_1, x_2, \ldots, x_n)$.



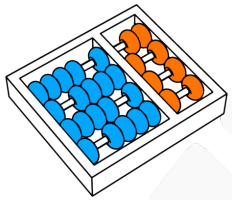
Reduzindo UE $\preccurlyeq_n$ PMP

1. Considere instância $I_{UE} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância $I_{PMP} = \{(x_1, 0), (x_2, 0), \ldots, (x_n, 0)\}$.



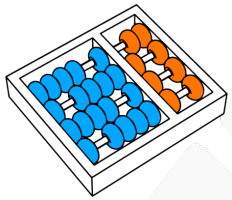
Reduzindo UE $\preceq_n$ PMP

1. Considere instância $I_{UE} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância $I_{PMP} = \{(x_1, 0), (x_2, 0), \ldots, (x_n, 0)\}$.



Reduzindo UE $\propto_n$ PMP

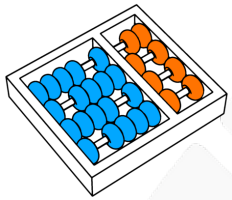
1. Considere instância $I_{UE} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância $I_{PMP} = \{(x_1, 0), (x_2, 0), \ldots, (x_n, 0)\}$.



Reduzindo UE $\preceq_n$ PMP

1. Considere instância $I_{UE} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância $I_{PMP} = \{(x_1, 0), (x_2, 0), \ldots, (x_n, 0)\}$.

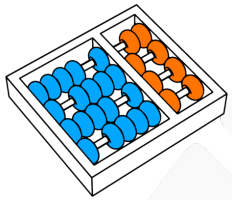
3. Resolva I_{PMP} e obtenha par de pontos $(x_i, 0), (x_j, 0)$.



Reduzindo UE $\preceq_n$ PMP

1. Considere instância $I_{UE} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância $I_{PMP} = \{(x_1, 0), (x_2, 0), \ldots, (x_n, 0)\}$.

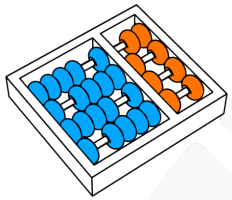
3. Resolva I_{PMP} e obtenha par de pontos $(x_i, 0), (x_j, 0)$.



Reduzindo UE $\preceq_n$ PMP

1. Considere instância $I_{UE} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância $I_{PMP} = \{(x_1, 0), (x_2, 0), \ldots, (x_n, 0)\}$.

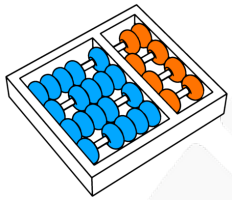
3. Resolva I_{PMP} e obtenha par de pontos $(x_i, 0), (x_j, 0)$.
4. Calcule a **DISTÂNCIA** d entre os pontos:



Reduzindo UE $\preccurlyeq_n$ PMP

1. Considere instância $I_{UE} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância $I_{PMP} = \{(x_1, 0), (x_2, 0), \ldots, (x_n, 0)\}$.

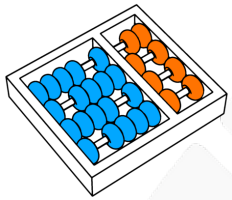
3. Resolva I_{PMP} e obtenha par de pontos $(x_i, 0), (x_j, 0)$.
4. Calcule a **DISTÂNCIA** d entre os pontos:
   (a) Se $d = 0$, então devolva NAO.



Reduzindo UE $\preccurlyeq_n$ PMP

1. Considere instância $I_{UE} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância $I_{PMP} = \{(x_1, 0), (x_2, 0), \ldots, (x_n, 0)\}$.

3. Resolva I_{PMP} e obtenha par de pontos $(x_i, 0), (x_j, 0)$.
4. Calcule a **DISTÂNCIA** d entre os pontos:
   (a) Se $d = 0$, então devolva NAO.
   (b) Se $d > 0$, devolva SIM.

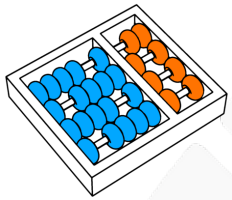


Reduzindo UE $\preceq_n$ PMP

1. Considere instância $I_{UE} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância $I_{PMP} = \{(x_1, 0), (x_2, 0), \ldots, (x_n, 0)\}$.

3. Resolva I_{PMP} e obtenha par de pontos $(x_i, 0), (x_j, 0)$.
4. Calcule a **DISTÂNCIA** d entre os pontos:
   (a) Se $d = 0$, então devolva NAO.
   (b) Se $d > 0$, devolva SIM.

- O tempo da redução é $O(n)$.



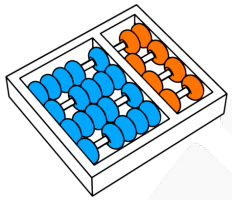
Reduzindo UE $\preccurlyeq_n$ PMP

1. Considere instância $I_{UE} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância $I_{PMP} = \{(x_1, 0), (x_2, 0), \ldots, (x_n, 0)\}$.

3. Resolva I_{PMP} e obtenha par de pontos $(x_i, 0), (x_j, 0)$.
4. Calcule a **DISTÂNCIA** d entre os pontos:
   (a) Se $d = 0$, então devolva NAO.
   (b) Se $d > 0$, devolva SIM.

- O tempo da redução é $O(n)$.
- Portanto $\Omega(n \log n)$ também é **COTA INFERIOR** para PMP.

3-Soma para Colinearidade

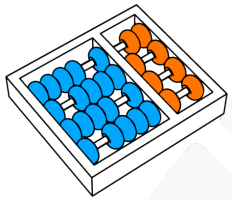


Problema de origem

Problema (3-Soma (3SUM))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *reais.*
Saída: *Determinar se existem índices distintos* i,j *e* k *tais que:*

$$x_i + x_j + x_k = 0.$$



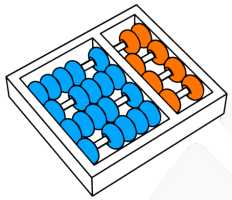
Problema de origem

Problema (3-Soma (3SUM))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *reais.*
Saída: *Determinar se existem índices distintos* i,j *e* k *tais que:*

$$x_i + x_j + x_k = 0.$$

Exemplo:



Problema de origem

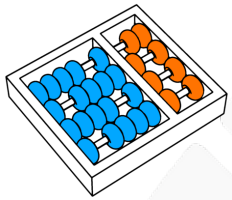
Problema (3-Soma (3SUM))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *reais.*
Saída: *Determinar se existem índices distintos* i,j *e* k *tais que:*

$$x_i + x_j + x_k = 0.$$

Exemplo:

- Instância $X = (\underline{4}, -6, \underline{1}, 8, 7, \underline{-5})$



Problema de origem

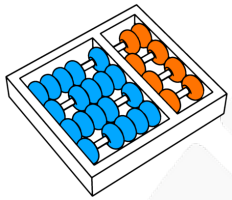
Problema (3-Soma (3SUM))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *reais.*
Saída: *Determinar se existem índices distintos* i,j *e* k *tais que:*

$$x_i + x_j + x_k = 0.$$

Exemplo:

- Instância $X = (\underline{4}, -6, \underline{1}, 8, 7, \underline{-5})$
- Solução $i = 1$, $j = 3$ e $k = 6$



Problema de origem

Problema (3-Soma (3SUM))

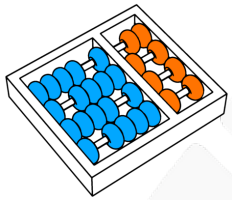
Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *reais.*
Saída: *Determinar se existem índices distintos* i,j *e* k *tais que:*

$$x_i + x_j + x_k = 0.$$

Exemplo:

- Instância $X = (\underline{4}, -6, \underline{1}, 8, 7, \underline{-5})$
- Solução $i = 1$, $j = 3$ e $k = 6$

Observações:



Problema de origem

Problema (3-Soma (3SUM))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *reais.*
Saída: *Determinar se existem índices distintos* i,j *e* k *tais que:*

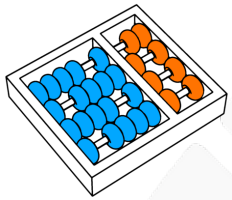
$$x_i + x_j + x_k = 0.$$

Exemplo:

- Instância $X = (\underline{4}, -6, \underline{1}, 8, 7, \underline{-5})$
- Solução $i = 1$, $j = 3$ e $k = 6$

Observações:

- Pode ser resolvido em $O(n^2)$.



Problema de origem

Problema (3-Soma (3SUM))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *reais.*
Saída: *Determinar se existem índices distintos* i,j *e* k *tais que:*

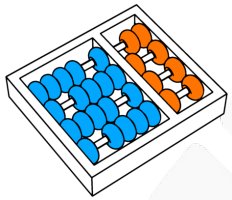
$$x_i + x_j + x_k = 0.$$

Exemplo:

- Instância $X = (\underline{4}, -6, \underline{1}, 8, 7, \underline{-5})$
- Solução $i = 1$, $j = 3$ e $k = 6$

Observações:

- Pode ser resolvido em $O(n^2)$ (como?).



Problema de origem

Problema (3-Soma (3SUM))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *reais.*
Saída: *Determinar se existem índices distintos* i,j *e* k *tais que:*

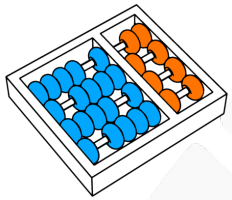
$$x_i + x_j + x_k = 0.$$

Exemplo:

- Instância $X = (\underline{4}, -6, \underline{1}, 8, 7, \underline{-5})$
- Solução $i = 1$, $j = 3$ e $k = 6$

Observações:

- Pode ser resolvido em $O(n^2)$ (como?).
- Acreditava-se que $\Omega(n^2)$ era **COTA INFERIOR**.



Problema de origem

Problema (3-Soma (3SUM))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *reais.*
Saída: *Determinar se existem índices distintos* i,j *e* k *tais que:*

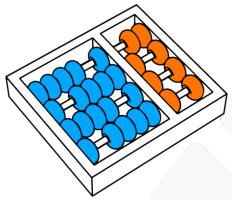
$$x_i + x_j + x_k = 0.$$

Exemplo:

- Instância $X = (\underline{4}, -6, \underline{1}, 8, 7, \underline{-5})$
- Solução $i = 1$, $j = 3$ e $k = 6$

Observações:

- Pode ser resolvido em $O(n^2)$ (como?).
- Acreditava-se que $\Omega(n^2)$ era **COTA INFERIOR**.
- Pode ser resolvido em $o(n^2)$ (Grønlund e Pettie, 2014).



Problema de origem

Problema (3-Soma (3SUM))

Entrada: *Uma sequência* $X = (x_1, x_2, \ldots, x_n)$ *de* n *reais.*
Saída: *Determinar se existem índices distintos* i,j *e* k *tais que:*

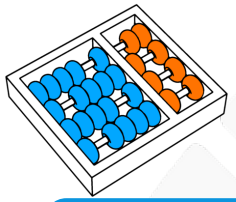
$$x_i + x_j + x_k = 0.$$

Exemplo:

- Instância $X = (\underline{4}, -6, \underline{1}, 8, 7, \underline{-5})$
- Solução $i = 1$, $j = 3$ e $k = 6$

Observações:

- Pode ser resolvido em $O(n^2)$ (como?).
- Acreditava-se que $\Omega(n^2)$ era **COTA INFERIOR**.
- Pode ser resolvido em $o(n^2)$ (Grønlund e Pettie, 2014).
- Ainda se acredita que não dá pra fazer melhor que $n^{2-\Omega(1)}$.

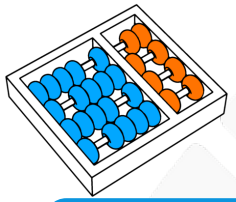


Problema de destino

Problema (Colinearidade Não Horizontal (COL))

Entrada: *Um conjunto* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de* n *pontos no plano.*

Saída: *Determinar se três pontos estão em alguma* ***RETA NÃO HORIZONTAL****.*

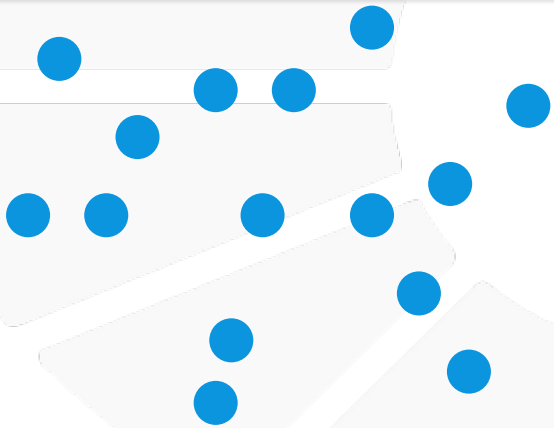


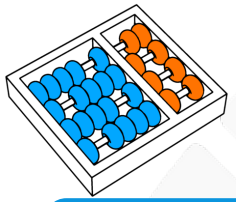
Problema de destino

Problema (Colinearidade Não Horizontal (COL))

Entrada: *Um conjunto* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de* n *pontos no plano.*

Saída: *Determinar se três pontos estão em alguma* ***RETA NÃO HORIZONTAL****.*



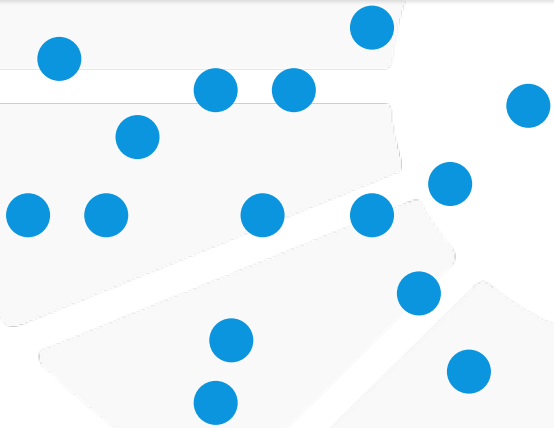


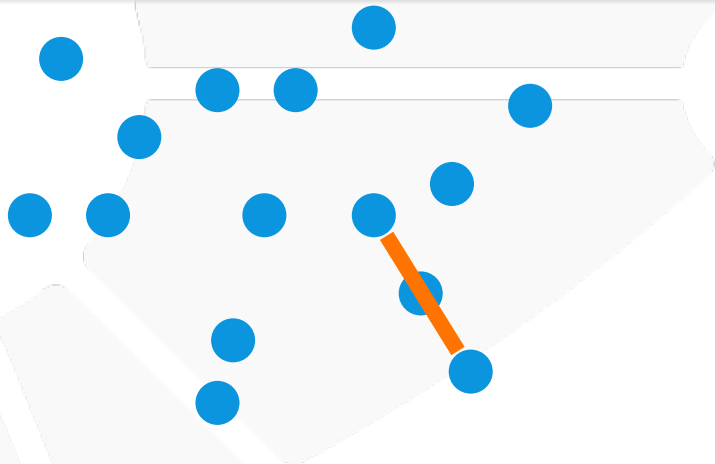
Problema de destino

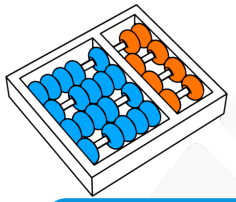
Problema (Colinearidade Não Horizontal (COL))

Entrada: *Um conjunto* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de* n *pontos no plano.*

Saída: *Determinar se três pontos estão em alguma* ***RETA NÃO HORIZONTAL****.*





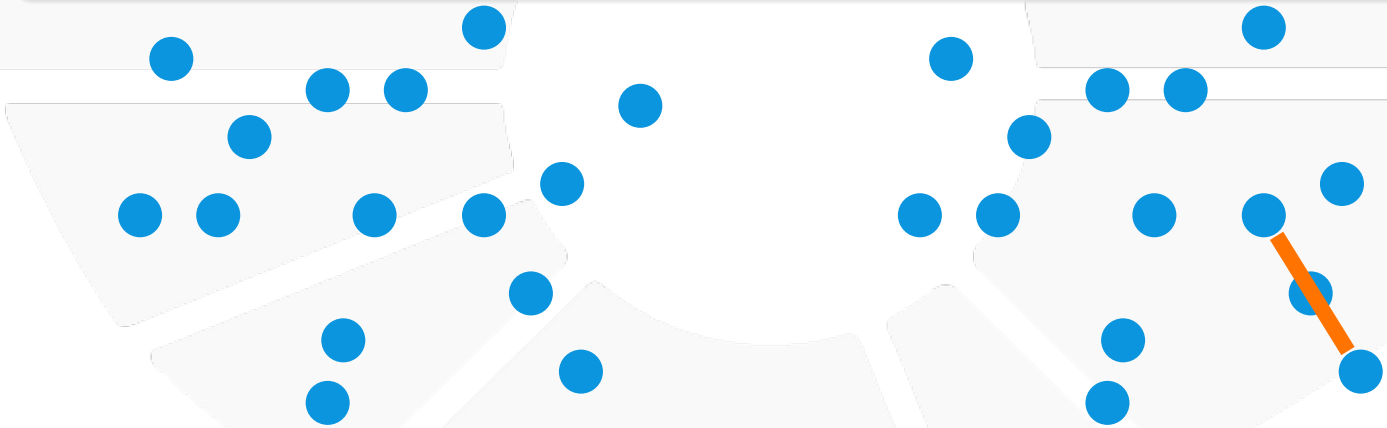


Problema de destino

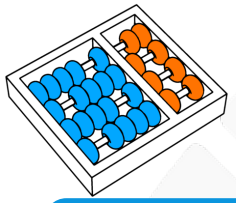
Problema (Colinearidade Não Horizontal (COL))

Entrada: *Um conjunto* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de n pontos no plano.*

Saída: *Determinar se três pontos estão em alguma* ***RETA NÃO HORIZONTAL****.*



Observações:

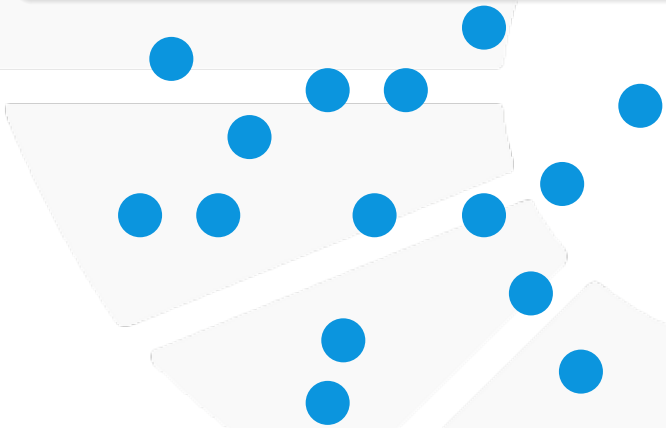


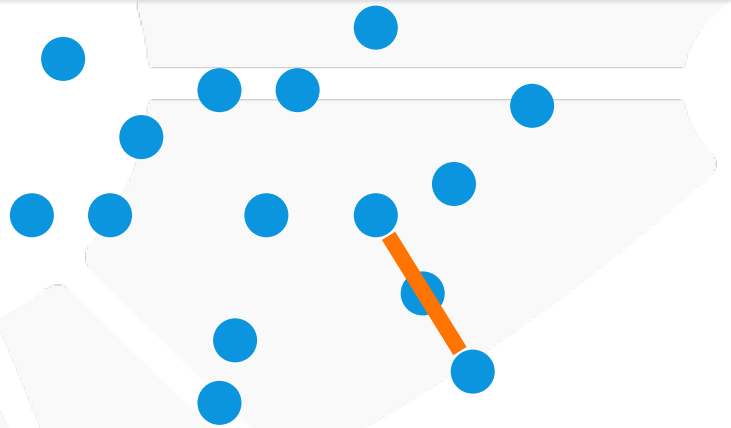
Problema de destino

Problema (Colinearidade Não Horizontal (COL))

Entrada: *Um conjunto* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de* n *pontos no plano.*

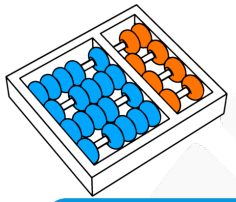
Saída: *Determinar se três pontos estão em alguma* ***RETA NÃO HORIZONTAL****.*





Observações:

- Pode ser resolvido em tempo $O(n^2)$.

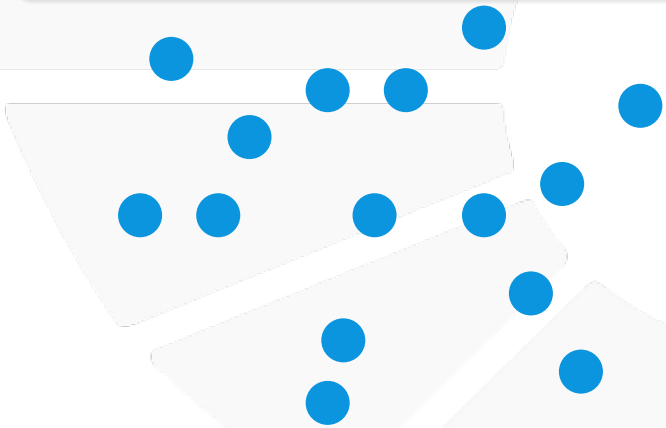


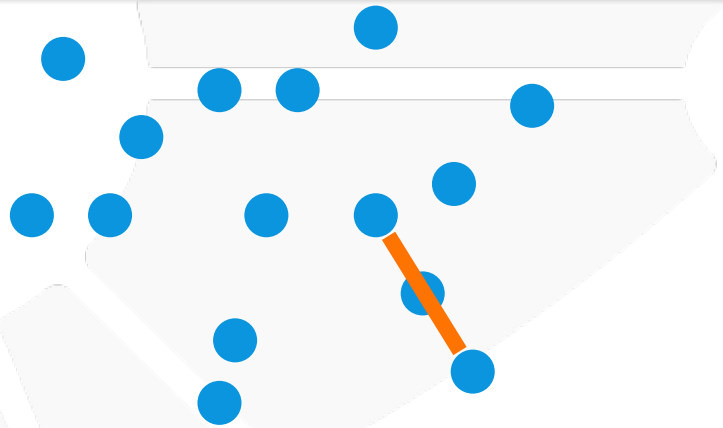
Problema de destino

Problema (Colinearidade Não Horizontal (COL))

Entrada: *Um conjunto* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *de* n *pontos no plano.*

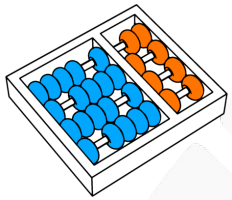
Saída: *Determinar se três pontos estão em alguma* ***RETA NÃO HORIZONTAL****.*





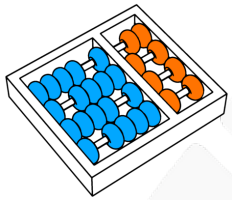
Observações:

- Pode ser resolvido em tempo $O(n^2)$.
- Acredita-se que $\Omega(n^2)$ é **COTA INFERIOR**.



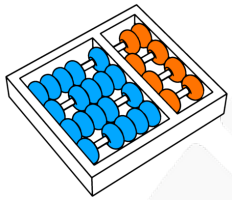
Reduzindo 3SUM $\prec_n$ COL

1. Considere instância $I_{3SUM} = (x_1, x_2, \ldots, x_n)$.



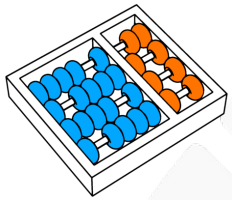
Reduzindo 3SUM $\preceq_n$ COL

1. Considere instância $I_{3SUM} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância
   $I_{COL} = \{(x_i, 0), (-x_i/2, 1), (x_i, 2) : i = 1, 2, \ldots, n\}$.



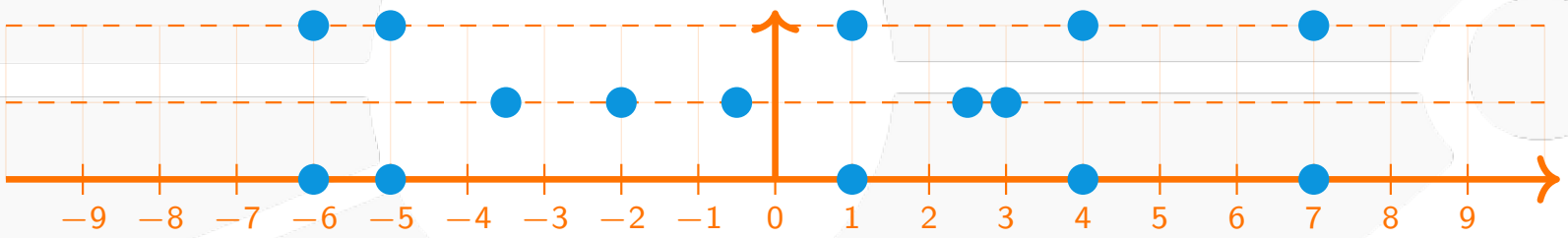
Reduzindo 3SUM $\preceq_n$ COL

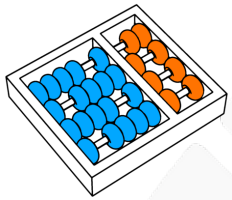
1. Considere instância $I_{3SUM} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância
   $I_{COL} = \{(x_i, 0), (-x_i/2, 1), (x_i, 2) : i = 1, 2, \ldots, n\}$.
   Exemplo: $X = (4, -6, 1, 8, 7, -5)$:



Reduzindo 3SUM $\prec_n$ COL

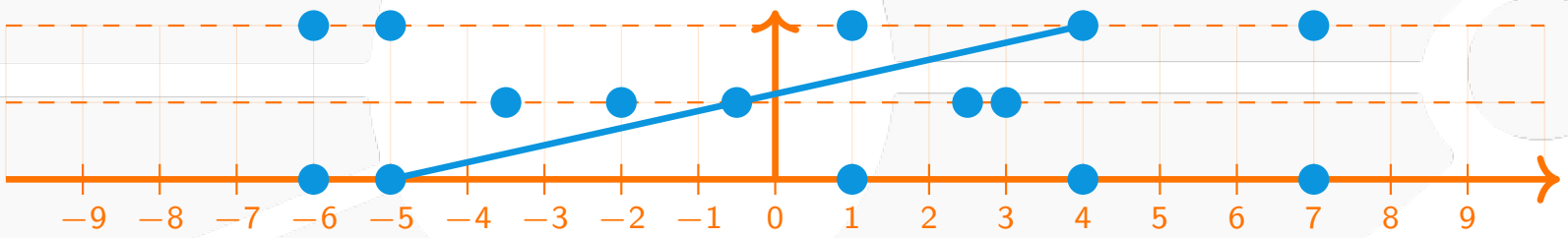
1. Considere instância $I_{3SUM} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância
   $I_{COL} = \{(x_i, 0), (-x_i/2, 1), (x_i, 2) : i = 1, 2, \ldots, n\}$.
   Exemplo: $X = (4, -6, 1, 8, 7, -5)$:



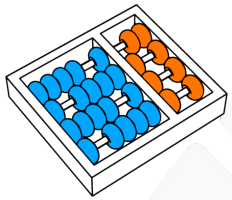


Reduzindo 3SUM $\preceq_n$ COL

1. Considere instância $I_{3SUM} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância
   $I_{COL} = \{(x_i, 0), (-x_i/2, 1), (x_i, 2) : i = 1, 2, \ldots, n\}$.
   Exemplo: $X = (4, -6, 1, 8, 7, -5)$:

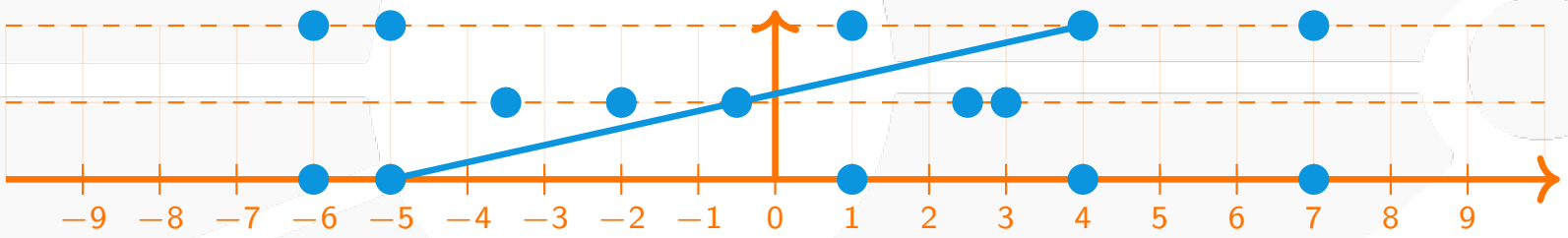


3. Resolva I_{COL} e obtenha S_{COL}.

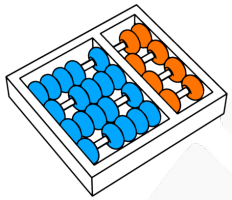


Reduzindo 3SUM $\preceq_n$ COL

1. Considere instância $I_{3SUM} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância
   $I_{COL} = \{(x_i, 0), (-x_i/2, 1), (x_i, 2) : i = 1, 2, \ldots, n\}$.
   Exemplo: $X = (4, -6, 1, 8, 7, -5)$:



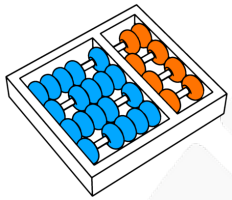
3. Resolva I_{COL} e obtenha S_{COL}.
4. Se a resposta S_{COL} for SIM, responda SIM,
   e se a resposta S_{COL} for NAO, responda NAO.



3SUM $\preccurlyeq_n$ COL (cont)

- A solução de I_{COL} (se houver) é uma tripla de pontos colineares. Claramente, cada um desses pontos deve estar em um dos eixos horizontais. Ou seja, tem a forma:

$$(x_i, 0), (-x_j/2, 1), (x_k, 2).$$

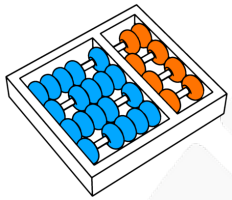


3SUM $\preccurlyeq_n$ COL (cont)

- A solução de I_{COL} (se houver) é uma tripla de pontos colineares. Claramente, cada um desses pontos deve estar em um dos eixos horizontais. Ou seja, tem a forma:

$$(x_i, 0), (-x_j/2, 1), (x_k, 2).$$

Portanto, $x_i + x_j + x_k = 0$.



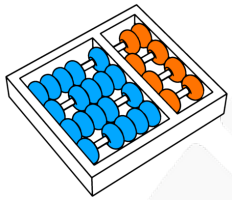
3SUM $\preccurlyeq_n$ COL (cont)

- A solução de I_{COL} (se houver) é uma tripla de pontos colineares. Claramente, cada um desses pontos deve estar em um dos eixos horizontais. Ou seja, tem a forma:

$$(x_i, 0), (-x_j/2, 1), (x_k, 2).$$

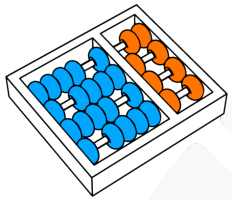
  Portanto, $x_i + x_j + x_k = 0$.

- Se $x_i + x_j + x_k = 0$, então, os pontos citados são colineares.



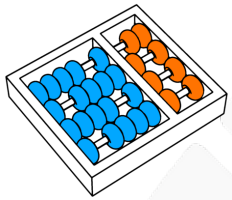
3SUM $\prec_n$ COL

- É claro que $\Omega(n)$ é uma cota inferior para 3SUM.



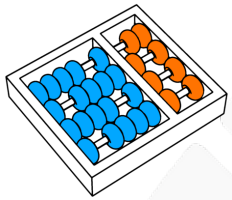
3SUM $\prec_n$ COL

- É claro que $\Omega(n)$ é uma cota inferior para 3SUM.
- E **SE** houver cota inferior $\Omega(h(n))$ maior para 3SUM?



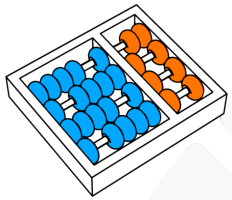
3SUM $\prec_n$ COL

- É claro que $\Omega(n)$ é uma cota inferior para 3SUM.
- E **SE** houver cota inferior $\Omega(h(n))$ maior para 3SUM?
  - A redução gasta tempo $f(n) = O(n)$.



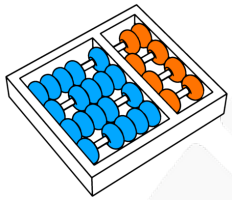
3SUM $\prec_n$ COL

- É claro que $\Omega(n)$ é uma cota inferior para 3SUM.
- E **SE** houver cota inferior $\Omega(h(n))$ maior para 3SUM?
  - A redução gasta tempo $f(n) = O(n)$.
  - Nesse caso, $f(n) = o(h(n))$.



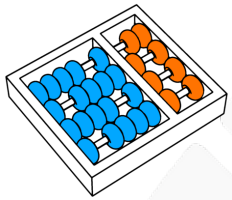
3SUM $\prec_n$ COL

- É claro que $\Omega(n)$ é uma cota inferior para 3SUM.
- E **SE** houver cota inferior $\Omega(h(n))$ maior para 3SUM?
  - A redução gasta tempo $f(n) = O(n)$.
  - Nesse caso, $f(n) = o(h(n))$.
  - Então, $\Omega(h(n))$ seria cota inferior para COL



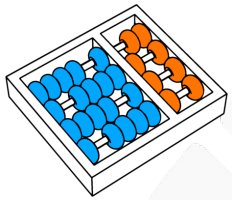
3SUM $\prec_n$ COL

- É claro que $\Omega(n)$ é uma cota inferior para 3SUM.
- E **SE** houver cota inferior $\Omega(h(n))$ maior para 3SUM?
  - A redução gasta tempo $f(n) = O(n)$.
  - Nesse caso, $f(n) = o(h(n))$.
  - Então, $\Omega(h(n))$ seria cota inferior para COL
- Mas só conhecemos a cota trivial $\Omega(n)$ para 3SUM.



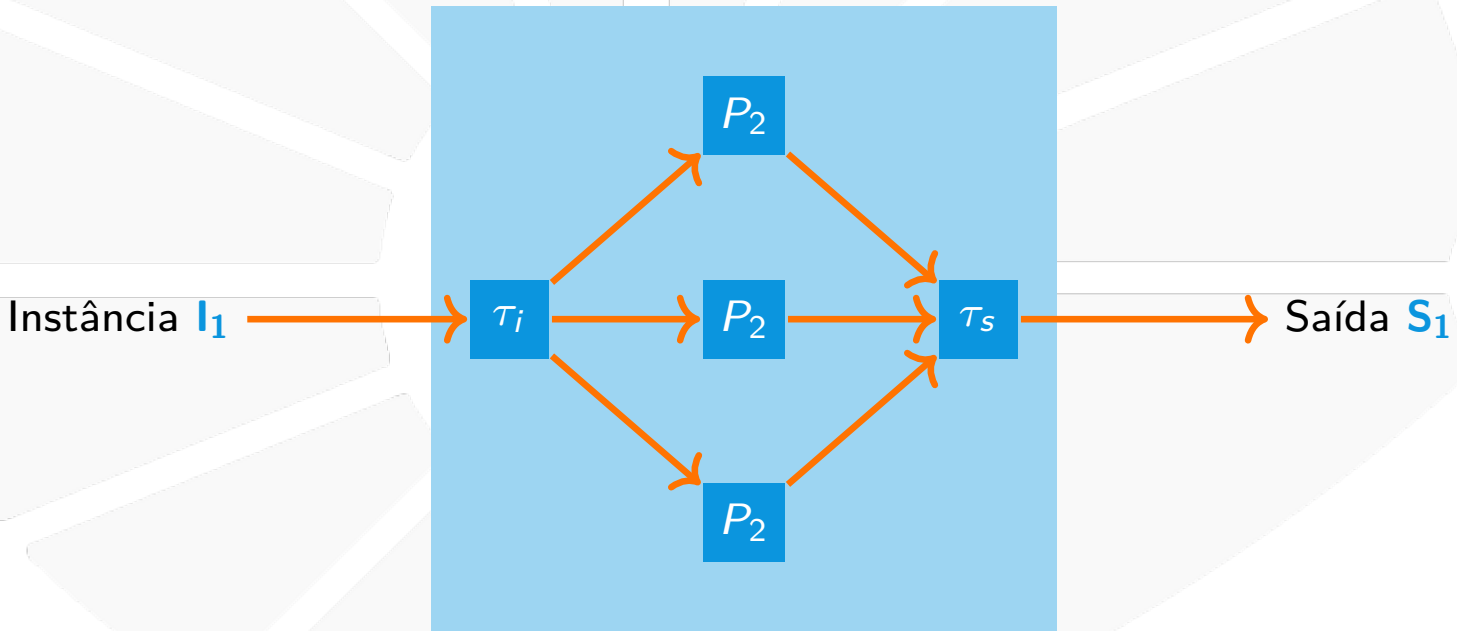
Redução de Turing

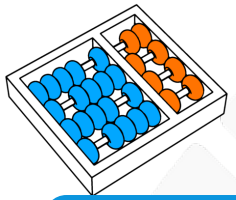
Podemos reduzir P_1 para P_2 fazendo várias aplicações de P_2.



Redução de Turing

Podemos reduzir P_1 para P_2 fazendo várias aplicações de P_2.



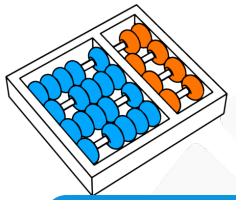


Exemplo de redução de Turing

Problema (Multiplicação de Inteiros)

Entrada: *Dois inteiros a e b.*

Saída: *O produto $a \cdot b$.*



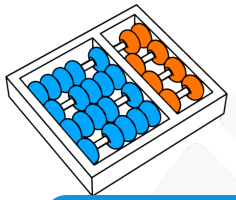
Exemplo de redução de Turing

Problema (Multiplicação de Inteiros)

Entrada: *Dois inteiros a e b.*
Saída: *O produto $a \cdot b$.*

Problema (Quadrado)

Entrada: *Um inteiro x.*
Saída: *O quadrado x^2.*



Exemplo de redução de Turing

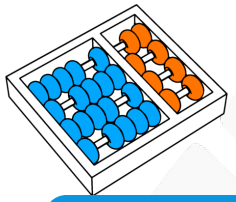
Problema (Multiplicação de Inteiros)

Entrada: *Dois inteiros a e b.*
Saída: *O produto $a \cdot b$.*

Problema (Quadrado)

Entrada: *Um inteiro x.*
Saída: *O quadrado x^2.*

Redução:



Exemplo de redução de Turing

Problema (Multiplicação de Inteiros)

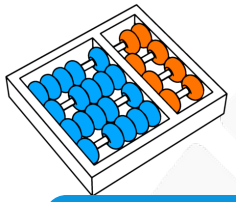
Entrada: *Dois inteiros a e b.*
Saída: *O produto $a \cdot b$.*

Problema (Quadrado)

Entrada: *Um inteiro x.*
Saída: *O quadrado x^2.*

Redução:

- Podemos reduzir Multiplicação de Inteiros para Quadrado.



Exemplo de redução de Turing

Problema (Multiplicação de Inteiros)

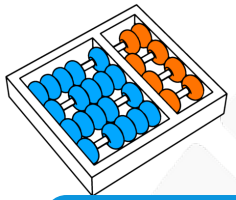
Entrada: *Dois inteiros a e b.*
Saída: *O produto $a \cdot b$.*

Problema (Quadrado)

Entrada: *Um inteiro x.*
Saída: *O quadrado x^2.*

Redução:

- Podemos reduzir Multiplicação de Inteiros para Quadrado.
- Fazemos apenas um número constante de somas, subtrações e divisão por dois



Exemplo de redução de Turing

Problema (Multiplicação de Inteiros)

Entrada: *Dois inteiros a e b.*
Saída: *O produto $a \cdot b$.*

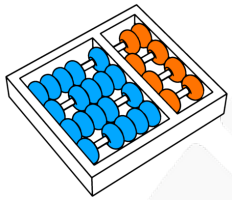
Problema (Quadrado)

Entrada: *Um inteiro x.*
Saída: *O quadrado x^2.*

Redução:

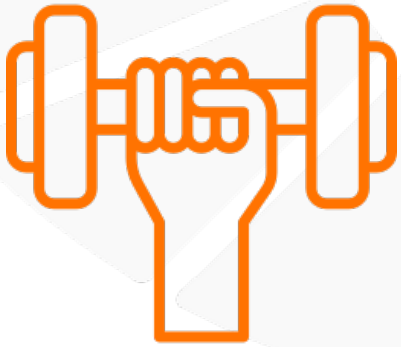
- Podemos reduzir Multiplicação de Inteiros para Quadrado.
- Fazemos apenas um número constante de somas, subtrações e divisão por dois

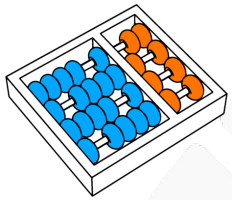
$$a \cdot b = \frac{(a+b)^2 - a^2 - b^2}{2}$$



Refletindo sobre reduções e cotas inferiores

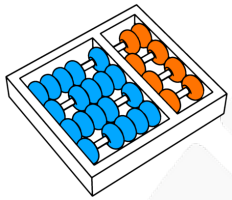
Vamos fazer alguns exercícios?





Exercício 1

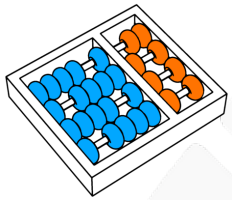
O problema 3SUMplus consiste em: dados uma sequência $X = (x_1, x_2, \ldots, x_n)$ de reais e um valor real b, determinar se existem três índices distintos i, j e k tais que $x_i + x_j + x_k = b$.



Exercício 1

O problema 3SUMplus consiste em: dados uma sequência $X = (x_1, x_2, \ldots, x_n)$ de reais e um valor real b, determinar se existem três índices distintos i, j e k tais que $x_i + x_j + x_k = b$.

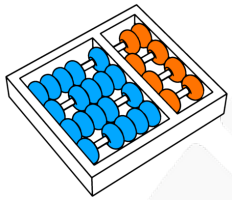
1. Mostre que 3SUM $\preccurlyeq_n$ 3SUMplus.



Exercício 1

O problema 3SUMplus consiste em: dados uma sequência $X = (x_1, x_2, \ldots, x_n)$ de reais e um valor real b, determinar se existem três índices distintos i, j e k tais que $x_i + x_j + x_k = b$.

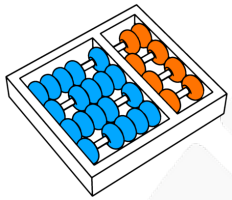
1. Mostre que 3SUM $\preccurlyeq_n$ 3SUMplus.
2. Mostre que 3SUMplus $\preccurlyeq_n$ 3SUM.



Exercício 1

O problema 3SUMplus consiste em: dados uma sequência $X = (x_1, x_2, \ldots, x_n)$ de reais e um valor real b, determinar se existem três índices distintos i, j e k tais que $x_i + x_j + x_k = b$.

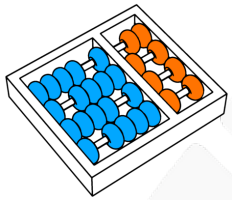
1. Mostre que 3SUM $\preccurlyeq_n$ 3SUMplus.
2. Mostre que 3SUMplus $\preccurlyeq_n$ 3SUM.
3. Suponha que o Professor Sabit Udo descobriu uma **COTA INFERIOR** de $\Omega(n^{1,9})$ para 3SUMplus. Nesse caso, quais das afirmações abaixo são verdadeiras?



Exercício 1

O problema 3SUMplus consiste em: dados uma sequência $X = (x_1, x_2, \ldots, x_n)$ de reais e um valor real b, determinar se existem três índices distintos i, j e k tais que $x_i + x_j + x_k = b$.

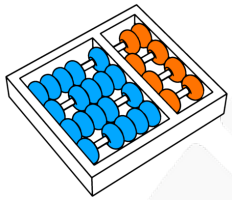
1. Mostre que 3SUM $\preccurlyeq_n$ 3SUMplus.
2. Mostre que 3SUMplus $\preccurlyeq_n$ 3SUM.
3. Suponha que o Professor Sabit Udo descobriu uma **COTA INFERIOR** de $\Omega(n^{1,9})$ para 3SUMplus. Nesse caso, quais das afirmações abaixo são verdadeiras?
   (i) Não existe algoritmo $O(n^{1,5})$ para 3SUMplus.



Exercício 1

O problema 3SUMplus consiste em: dados uma sequência $X = (x_1, x_2, \ldots, x_n)$ de reais e um valor real b, determinar se existem três índices distintos i, j e k tais que $x_i + x_j + x_k = b$.

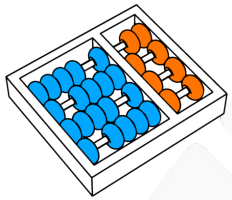
1. Mostre que 3SUM $\preccurlyeq_n$ 3SUMplus.
2. Mostre que 3SUMplus $\preccurlyeq_n$ 3SUM.
3. Suponha que o Professor Sabit Udo descobriu uma **COTA INFERIOR** de $\Omega(n^{1,9})$ para 3SUMplus. Nesse caso, quais das afirmações abaixo são verdadeiras?
   - (i) Não existe algoritmo $O(n^{1,5})$ para 3SUMplus.
   - (ii) Não existe algoritmo $O(n^{1,5})$ para 3SUM.



Exercício 1

O problema 3SUMplus consiste em: dados uma sequência $X = (x_1, x_2, \ldots, x_n)$ de reais e um valor real b, determinar se existem três índices distintos i, j e k tais que $x_i + x_j + x_k = b$.

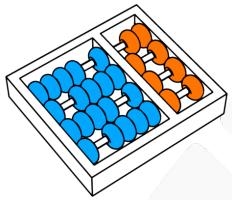
1. Mostre que 3SUM $\preccurlyeq_n$ 3SUMplus.
2. Mostre que 3SUMplus $\preccurlyeq_n$ 3SUM.
3. Suponha que o Professor Sabit Udo descobriu uma **COTA INFERIOR** de $\Omega(n^{1,9})$ para 3SUMplus. Nesse caso, quais das afirmações abaixo são verdadeiras?
   - (i) Não existe algoritmo $O(n^{1,5})$ para 3SUMplus.
   - (ii) Não existe algoritmo $O(n^{1,5})$ para 3SUM.
   - (iii) Existe um algoritmo $O(n^{1,9})$ para 3SUMplus.



Exercício 1

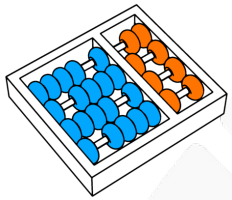
O problema 3SUMplus consiste em: dados uma sequência $X = (x_1, x_2, \ldots, x_n)$ de reais e um valor real b, determinar se existem três índices distintos i, j e k tais que $x_i + x_j + x_k = b$.

1. Mostre que 3SUM $\preccurlyeq_n$ 3SUMplus.
2. Mostre que 3SUMplus $\preccurlyeq_n$ 3SUM.
3. Suponha que o Professor Sabit Udo descobriu uma **COTA INFERIOR** de $\Omega(n^{1,9})$ para 3SUMplus. Nesse caso, quais das afirmações abaixo são verdadeiras?
   - (i) Não existe algoritmo $O(n^{1,5})$ para 3SUMplus.
   - (ii) Não existe algoritmo $O(n^{1,5})$ para 3SUM.
   - (iii) Existe um algoritmo $O(n^{1,9})$ para 3SUMplus.
   - (iv) Existe um algoritmo $O(n^{1,9})$ para 3SUM.



Exercício 2

Considere os problemas:



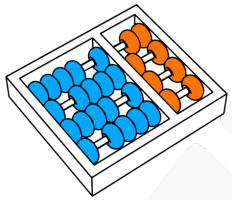
Exercício 2

Considere os problemas:

Problema (Sistema de Representantes Distintos (SRD))

Entrada: *Uma coleção de conjuntos* $S_1, \ldots, S_k$.

Saída: *Conjunto* $R = \{r_1, \ldots, r_k\}$ *tal que* $r_i \in S_i$ *para* $i = 1, \ldots, k$.



Exercício 2

Considere os problemas:

Problema (Sistema de Representantes Distintos (SRD))

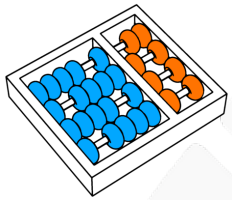
Entrada: *Uma coleção de conjuntos* $S_1, \ldots, S_k$.

Saída: *Conjunto* $R = \{r_1, \ldots, r_k\}$ *tal que* $r_i \in S_i$ *para* $i = 1, \ldots, k$.

Problema (Emparelhamento Máximo (EM))

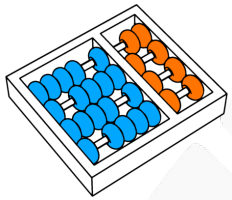
Entrada: *Um grafo bipartido* $G = (X \cup Y, E)$ *com bipartição* X *e* Y.

Saída: *Um subconjunto de arestas* M *que não compartilham vértices, tal que* $|M|$ *seja máximo.*



Exercício 2

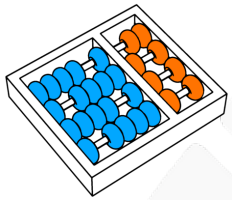
Mostre que SRD $\preccurlyeq$ EM.



Exercício 2

Mostre que SRD $\preccurlyeq$ EM.

Um fato útil, pode ser o seguinte teorema:



Exercício 2

Mostre que SRD $\preccurlyeq$ EM.

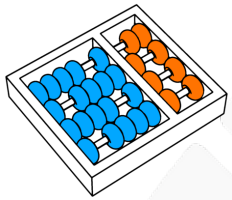
Um fato útil, pode ser o seguinte teorema:

Teorema (Teorema de Hall)

Uma coleção $S_1, \ldots, S_k$ tem um SRD se e somente se

$$\left|\{S_{i_1} \cup \cdots \cup S_{i_m}\}\right| \geq m$$

para ***TODA*** *coleção $\{i_1, \ldots, i_m\} \subseteq \{1, 2, 3, \ldots, k\}$.*

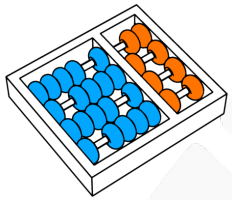


Exercício 3

Considere os seguintes problemas:

Problema (Edição de String)

Entrada: *Duas strings A e B.*
Saída: *Menor sequência de operações para transformar A em B, onde as possíveis operações são:* ***inserção*** *de um caractere,* ***remoção*** *de um caractere, ou* ***troca*** *de um caractere por outro.*



Exercício 3

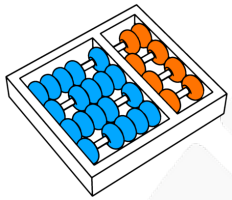
Considere os seguintes problemas:

Problema (Edição de String)

Entrada: *Duas strings A e B.*
Saída: *Menor sequência de operações para transformar A em B, onde as possíveis operações são:* ***inserção*** *de um caractere,* ***remoção*** *de um caractere, ou* ***troca*** *de um caractere por outro.*

Problema (Caminho Mínimo)

Entrada: *Um grafo direcionado $G(V, E)$, um peso $c_{ij} \geq 0$ para cada aresta $(i, j) \in E$, dois vértices s e t.*
Saída: *Um caminho de s a t em G de comprimento mínimo.*



Exercício 3

Considere os seguintes problemas:

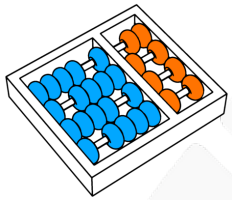
Problema (Edição de String)

Entrada: *Duas strings A e B.*
Saída: *Menor sequência de operações para transformar A em B, onde as possíveis operações são:* ***inserção*** *de um caractere,* ***remoção*** *de um caractere, ou* ***troca*** *de um caractere por outro.*

Problema (Caminho Mínimo)

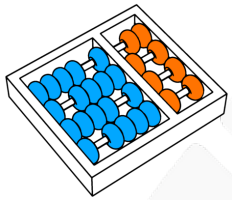
Entrada: *Um grafo direcionado $G(V, E)$, um peso $c_{ij} \geq 0$ para cada aresta $(i, j) \in E$, dois vértices s e t.*
Saída: *Um caminho de s a t em G de comprimento mínimo.*

Mostre como reduzir Edição de String a Caminho Mínimo.



Exercício 4

Considere os seguintes problemas:



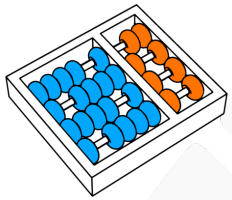
Exercício 4

Considere os seguintes problemas:

Problema (Ordenação)

Entrada: *Uma sequência de números naturais distintos* $x_1, x_2, \ldots, x_n$.

Saída: *Uma permutação ordenada dos números de entrada.*



Exercício 4

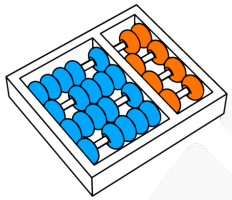
Considere os seguintes problemas:

Problema (Ordenação)

Entrada: *Uma sequência de números naturais distintos* $x_1, x_2, \ldots, x_n$.
Saída: *Uma permutação ordenada dos números de entrada.*

Problema (Codificação de Huffman)

Entrada: *Um alfabeto C e uma tabela de frequências f.*
Solução: *Uma codificação de comprimento variável que minimize o tamanho do texto codificado.*



Exercício 4

Considere os seguintes problemas:

Problema (Ordenação)

Entrada: *Uma sequência de números naturais distintos* $x_1, x_2, \ldots, x_n$.
Saída: *Uma permutação ordenada dos números de entrada.*

Problema (Codificação de Huffman)

Entrada: *Um alfabeto C e uma tabela de frequências f.*
Solução: *Uma codificação de comprimento variável que minimize o tamanho do texto codificado.*

Mostre como reduzir Ordeação para Codificação de Huffman.

Reduções e cotas inferiores

MC558 - Projeto e Análise de Algoritmos II

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

UNICAMP